VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro rozvoj kognitivních a motorických dovedností s využitím MonoGame Framework

Application for cognitive and motoric skills training based on MonoGame Framework

2015 Ondřej Gronych

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ondřej Gronych**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace pro rozvoj kognitivních a motorických dovedností s využitím MonoGame Framework
Application for Cognitive and Motoric Skills Training Based on MonoGame Framework

Zásady pro vypracování:

MonoGame Framework je Open Source implementace vývojářské platformy XNA 4.0 společnosti umožňující snadný vývoj her v jazyce C#. Cílem této práce je vyvinout pomocí MonoGame Frameworku aplikaci pomáhající rozvoji kognitivních a motorických dovedností formou nácviku a opakování jednoduchých cviků a úkolů.

1. Rešerše počítačových aplikací užívaných pro kompenzaci motorických a mentálních handicapů.
2. Výběr vhodných metod a forem tréninků kognitivních a motorických funkcí uživatelů.
3. Implementace zvolených metod na operačním systému, který podporuje technologii MonoGame Framework.
4. Tvorba uživatelského rozhraní a vykreslování grafiky pomocí MonoGame Framework.
5. Testování aplikace včetně výkonnostních testů.

Seznam doporučené odborné literatury:

[1] Microsoft MSDN Library - http://msdn.microsoft.com/en-US/
[2] XNA Developer Center - http://msdn.microsoft.com/en-us/aa937791.aspx

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 11. března 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. března 2015 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Jmenovitě pak vedoucímu mé diplomové práce Ing. Janu Martinovičovi, Ph.D za jeho cenné rady a postřehy. Také bych rád poděkoval panu Ing. Jiřímu Krajíčkovi za jeho spolupráci.

Abstrakt

Tato práce se zabývá vývojem aplikace pro rozvoj kognitivních a motorických dovedností jedince pomocí grafického frameworku MonoGame. V práci je tento framework blíže rozebrán a jsou v ní vysvětleny základní principy pro práci s ním. Pomocí tohoto frameworku byla implementována aplikace, která byla poté testována dětmi s lehkou a střední mentální retardací. Výsledkem testování bylo zhodnocení použitelnosti aplikace. V závěru práce jsou poté uvedeny výsledky výkonnostních testů aplikace.

Klíčová slova: MonoGame, XNA, kognitivní funkce, mobilní aplikace, Windows Store

Abstract

The aim of the thesis is the implementation of a software, that improves development of cognitive and motoric skills of individuals, using a graphical framework MonoGame. Thesis contains description of the framework and basic principles of working with it. Application, that was created using MonoGame framework was then tested on children with light to medium mental retardations. The result of the testing was to evaluate the usibility of the application. The last part of the thesis contains an interpretation of values measured during performance tests.

Keywords: MonoGame, XNA, cognitive skills, mobile application, Windows Store

Seznam použitých zkratek a symbolů

WACK	–	Windows App Certification Kit
XAML	–	Extensible Application Markup Language
HTML	–	Hyper Text Markup Language
XML	–	Extensible Markup Language
SEN	–	Special Education Needs
DOM	–	Document Object Model
DCC	–	Digital Content Creation
MSDN	–	Microsoft Developer Network
HCI	–	Human Computer Interaction
GUI	–	Graphical User Interface
FPS	–	Frame Per Second

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

Obsah

1 Úvod

V současné době trh mobilních aplikací obsahuje širokou škálu aplikací s nejrůznějším zaměřením. Aplikace mohou být určené například ke každodennímu použití (počasí, kalendář...), k trávení volného času (například nejrůznější hry), nebo mimo jiné také k rozvoji uživatele a právě tímto směrem se ubírá tato práce. Konkrétně vývojem aplikace pro rozvoj různých kognitivních schopností a jemné motoriky. K tomuto účelu byl využit grafický framework MonoGame.

V úvodu (kapitola č. 2) se práce zabývá aktuálním stavem podobných aplikací na současném trhu. Jsou zde vybrány nejzajímavější aplikace a ty jsou krátce představeny, především jejich výhody a nevýhody. Dále v kapitole č. 3: se práce zmiňuje o zvoleném operačním systému, kterým byl Windows 8, konkrétně jeho tzv. metro prostředí. To především z důvodu možnosti bezplatného vývoje pro tuto platformu s využitím frameworku MonoGame. O samotném frameworku se zmiňuje kapitola 4. V této části práce jsou popsány rozdíly mezi verzemi, které ovlivnily vývoj aplikace. V počátcích totiž byla používána starší verze než na konci vývoje a právě o důležitých změnách, které s sebou nová verze přinesla se tato kapitola zmiňuje. Dále se lze zde dočíst o multiplatformnosti tohoto frameworku, herní smyčce, nebo o content pipeline. Následující kapitola č. 5 popisuje samotnou implementaci aplikace. V úvodu této sekce jsou obsaženy obecné informace o vyvíjené aplikaci (z jakých úloh se skládá apod.), dále následují informace o 2D a 3D implementaci, o implementaci kolizí objektů, nebo například zvukových efektů. V těchto dvou částech jsou uvedeny jak obecné informace o frameworku, tak i konkrétní ukázky implementace. Cílem těchto dvou kapitol (č. 4 a č. 5) je seznámit čtenáře s výše uvedeným frameworkem a po přečtení by již měl zvládat jeho základy pro případnou práci s ním. Dále následuje kapitola č. 6, která je o tzv. použitelnosti a obsahuje informace o tom, co to vlastně použitelnost je a k čemu slouží. Tato použitelnost byla reálně testována dětmi s lehkou a střední mentální retardací. Výsledky tohoto testování jsou také uvedeny v této kapitole společně s informacemi o přínosu testování pro vyvíjenou aplikaci. Poslední kapitola č. 7 se zabývá výkonnostním testování aplikace. Jsou zde uvedeny například informace o rychlosti vykreslování na různých typech zařízení, za jakých okolností a proč dochází ke snížení fps (snímků zobrazených za sekundu) apod.

2 Motorické a mentální hendikepy

Mentální hendikep jinak taky mentální retardace je postižení jedinců, při kterém zaostává vývoj rozumových schopností. Mentální retardace představuje výrazně sníženou úroveň inteligence a dle její velikosti se mentální retardace dělí do šesti kategorií [1]. Označení "mentální hendikep" platí pouze pro některé schopnosti jedince, jiné mohou být normální, nebo až nadprůměrné. Primárním specifikem mentální retardace je postižení kognitivních (poznávacích) funkcí jedince. Bohužel pouhé zařazení do kategorie dle IQ nestačí k dobrému porozumění chování jedince s mentální retardací a jeho rozvoji. K tomu je potřeba znát silné a slabé stránky jednotlivých kognitivních funkcí jako jsou například paměť, pozornost, percepční schopnosti apod.

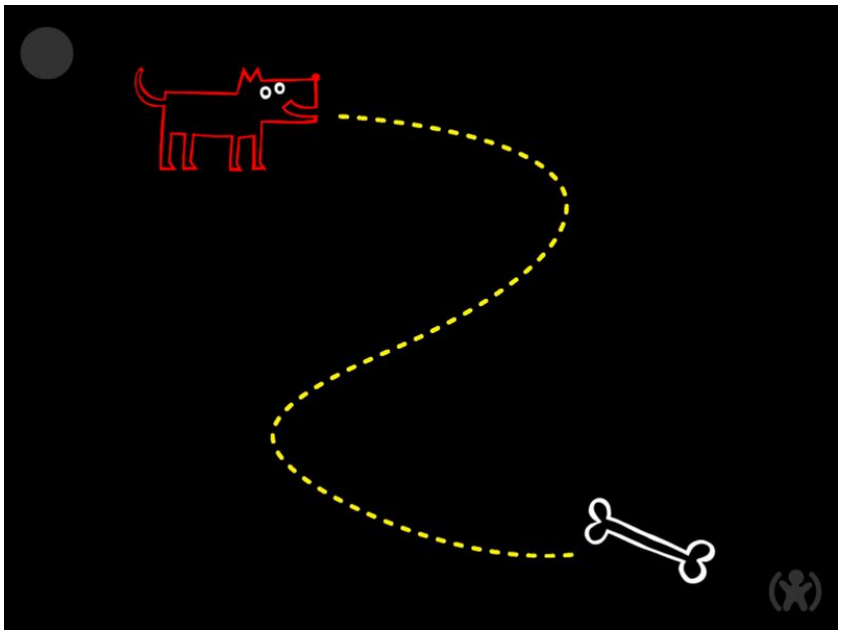
2.1 Aplikace užívané pro rozvoj motorických a kognitivních funkcí

V rámci diplomové práce byl proveden průzkum současného stavu aplikací pro rozvoj kognitivních a motorických funkcí. Aplikace na současném trhu mají různá zaměření, avšak malý zlomek z nich je určen právě pro rozvoj kognitivních funkcí. Většina aplikací, které jsou určeny pro děti a jedince s určitým mentálním hendikepem se zaměřují na výuku. Ať už se jedná o zlepšování čtení, psaní, mluvení, nebo například poznávání objektů reálného světa. Jelikož se jedná o aplikace určené na dotyková zařízení, lze naprostou většinu z nich považovat za rozvojové v oblasti jemné motoriky. Při hledání aplikace zaměřující se na rozvoj kognitivních schopností jedince může nastat hned několik problémů. Může se jednat o jazykovou bariéru (většina aplikací jsou v angličtině, či jiných jazycích), většina jich je placená apod. Z důvodu nedostatku aplikací určených pro rozvoj kognitivních funkcí ve Windows Store byly zkoumány i aplikace určené pro jiné platformy (Android a iOS).

2.1.1 Eda Play

Jedná se o placenou českou aplikaci určenou pro platformu iOS. Její aktuální cena je 4,99€ a obsahuje přes 40 úkolů a 4 úrovně obtížnosti. Vytvořila ji nezisková organizace Raná péče EDA a jejím zakoupením putují peníze právě do této organizace na financování jejich projektů. Více informací o aplikaci lze nalézt na stránkách aplikace www.edaplay.cz [2] a také většina informací obsažených v této sekci byly čerpány z těchto webových stránek. Hlavním cílem aplikace Eda Play je pomáhat dětem trénovat zrak a jemnou motoriku. Není ale určená jenom pro děti s tímto hendikepem, je vhodná i pro zcela zdravé děti a využívat jí mohou již od jednoho roku věku [3]. Pokud se například dítě narodí se zakalenými očními

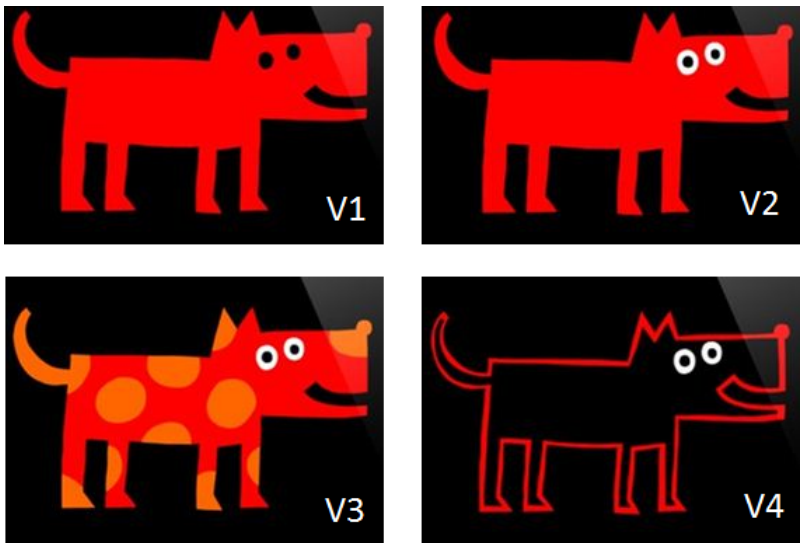
čočkami, následuje chirurgický zásah, který je ale pouze nutným předpokladem pro zlepšení. Rozdíl mezi cvičeným a necvičeným okem může být obrovský. "Zatímco necvičené oko zůstane na úrovni 10% normálního vidění, cvičené oko podle mých zkušeností může dosáhnout cca 50 - 100% normálního vidění", vysvětlovala paní Markéta Skalická, která je mimo jiné instruktorkou zrakové stimulace a metodik pro rozvoj zrakového vnímání a zrakovou terapeutkou Centra zrakových vad FN Motol [3]. Aplikace byla vyvíjena ve spolupráci s předními odborníky (mimo jiné i s paní Skalickou), a proto jednotlivé úlohy nejsou nahodilé, ale každá má přesně daný účel (v aplikaci se vyskytují aplikace typu dotknout se smutného obličeje, aby se stal veselým, přesunout letadlo k mraku apod.).



Obrázek 1: Ukázka úkolu z aplikace Eda Play [2]

Znění pro úlohu na výše uvedeném obrázku č. 1 je: "Pejsek má rád kosti. Doveď pejska po cestě ke kosti.". Aby děti ale věděli co vlastně mají dělat, obsahuje aplikace audio průvodce (vlídný dětský hlas), který jim mimo jiné třeba i radí při plnění úkolů. Tento audio průvodce je namluven jak v češtině, tak i v angličtině. Jak již bylo zmíněno výše, aplikace obsahuje čtyři úrovně obtížnosti úkolů. Ta nejjednodušší úroveň reaguje už pouhým dotykem na displej (například "Kohout vzbudí i ty největší spáče. Dotkni se a uslyšíš, jak kokrhá."). V dalších úrovních pak úlohy vyžadují dotyk na konkrétní místo. Následně pak úlohy procvičují pohyb rukou určitým směrem a v nejtěžší kategorii dochází ke cvičení už složitějších tvarů (například "Auto potřebuje kola, aby mohlo jet. Domaluj autu kola."). Kromě čtyř úrovní obtížností aplikace obsahuje ještě čtyři různé

úrovně zrakové náročnosti, které se v aplikací uvádí od V1 do V4.



Obrázek 2: Obrázky aplikace Eda Play pro různé úrovně zrakové vady [2]

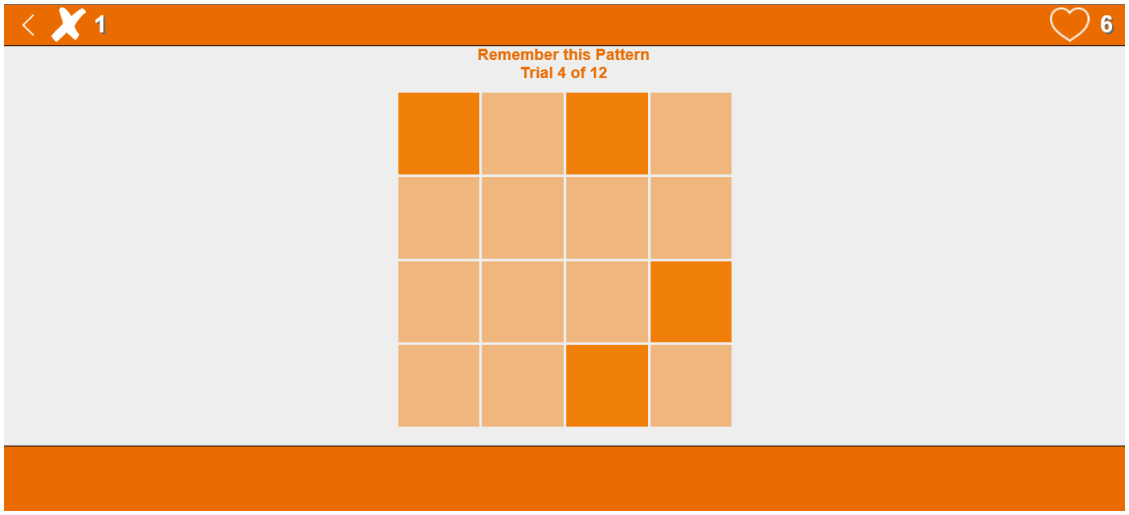
V1 je nejjednodušší zraková náročnost. Zobrazují se vždy plnobarevné obrázky, bez detailů. V2 přináší také plnobarevné obrázky, ale už s několika detaily. V3 pak nabízí obrázky s důrazem na několik detailů, často ve dvou barvách. V4 je nejnáročnější zraková úroveň, obrázky jsou zpracované jen jako obrysy.

2.1.2 Mind Games

Mind Games je multiplatformní aplikace vyvinutá společností Mindware Consulting, Inc určená pro tréning mozku. Na platformách Android, Windows Phone 8 a Windows 8.1 patří dokonce mezi nejlépe hodnocené aplikace v této kategorii. Mimo tyto platformy je aplikace přístupná ještě na platformě iOS a lze jí pustit i v prohlížeči na webových stránkách aplikace [1]. Dle autorů je hra vhodná pro uživatele od 12 let. Tato aplikace se vyskytuje na trhu ve dvou verzích. Jedna je zdarma a druhá verze tzv. Pro stojí aktuálně 100 Kč. Bezplatná verze je omezená pro některé dílčí úlohy (obsahuje pouze tři pokusy) a pro jejich zpřístupnění je potřeba plná verze. Aplikace obsahuje celkem 23 úloh pro cvičení mozku, rozdělených do různých kategorií (například mentální flexibilita, vizuální paměť, verbální paměť, prostorová paměť nebo třeba vědomosti).

Na uvedeném obrázku č. 3 je zobrazena ukázka hry z kategorie prostorové paměti. Všechny hry obsahují historii výsledků uživatele a graf pokroku. Aplikace využívá nor-

[1] webové stránky aplikace Mind Games: mindgames.mindware.mobi



Obrázek 3: Ukázka hry aplikace Mind Games [4]

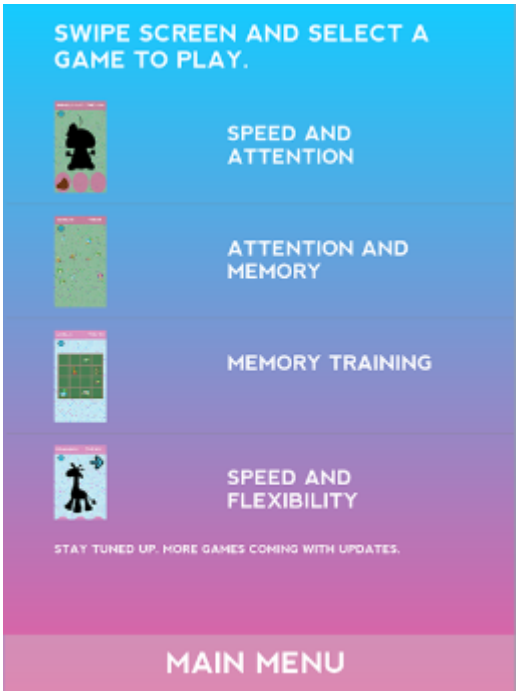
malizovaných testů a jejich výsledky jsou taktéž převedeny do normalizované podoby, takže uživatel na jejich základě ví, kde má nedostatky nebo naopak.

2.1.3 Cognitive Training

Celým názvem Cognitive Training on your Computer (zkráceně CoCo) je německá aplikace určená pro děti již od tří let a lze si jí stáhnou z Windows Storu [5]. Je k dispozici ve dvou jazycích a to v němčině a angličtině. Jedná se opět o souhrn dílčích úloh, několika obtížností pro trénování různých schopností (například vnímání, schopnost reagovat, jazykové schopnosti nebo manipulaci s čísly). Aplikaci je možné vyzkoušet zdarma, avšak tato verze obsahuje pouze tři úlohy z celkového počtu 17 úloh. Plná verze následně stojí 30 Kč. Všechny úlohy obsažené v této aplikaci byly vytvářeny s pomocí specializovaných odborníků v oblasti ergoterapie a dnes je využívána v mnoha terapeutických centrech, školách a nemocnicích.

2.1.4 Competitive Brain Training

Tuto aplikaci lze nalézt na Google play také pod názvem: "Chytráček: IQ testy pro děti" a to dokonce i s českým popisem aplikace, avšak po nainstalování není k dispozici jiný jazyk, než původní angličtina [6]. Aplikace je určena pro věkovou kategorii dětí od 4 let a je k dispozici pro platformy iOS a Android (na obou bezplatně). Zaměřuje se především na trénování paměti, pozornosti, rychlosti myšlení a logického myšlení.



Obrázek 4: Ukázka aplikace Competitive Brain Training

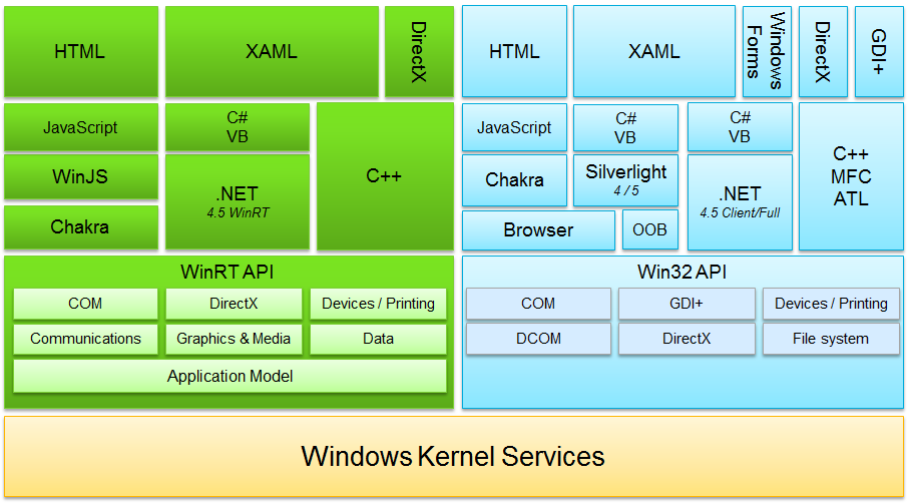
Po ukončení tréninku se zobrazí výsledné skóre a také výsledky z předchozích deseti tréninků. Je tak možné sledovat, zda-li dochází ke zlepšení. Aplikace navíc obsahuje i audio průvodce, bohužel ale jenom v anglickém jazyce.

3 Vývoj aplikací na Windows 8

Windows 8 přinesl novou koncepci zvaných Windows Store aplikací. Jedná se o celoobrazovkové aplikace, které běží v rámci nové nabídky Start. Uživatel si je může nainstalovat právě z vestavěného obchodu Windows Store.

3.1 Porovnání Windows Runtime a Windows Desktop

Výše uvedené aplikace běží nad úplně novým API zvaným WinRT nebo Windows Runtime, které přináší určitou náhradu či alternativu staršímu Win32 API [7]. Aplikace psané nad WinRT jsou optimalizované pro nízkou spotřebu baterie, není u nich například standardní multitasking. Model spouštění a uspávání aplikací na pozadí je podobný mobilním telefonům, kdy na displeji může být v jednu chvíli pouze jedna nebo maximálně dvě aplikace a ostatní jsou uspané na pozadí. Hlavním cílem je, aby vše dobře běželo i na slabých tabletech s procesory Atom nebo ARM a nespotřebovávalo moc baterie. Win32 API je nadále podporováno operačním systémem a proto všechny aplikace z Windows 7 normálně fungují i v desktopovém prostředí Windows 8.



Obrázek 5: Architektura Windows 8 [7]

V novém WinRT API se nachází poměrně hodně změn. WinRT je nativní neřízené (unmanaged) API založené na COM. Nad tímto API se nachází projekce jednotlivých jazyků, dostupný je nativní jazyk C a C++, jazyk C♯ VB.NET a aplikace je možné psát i v HTML a JavaScriptu. Jednotlivé aplikace napsané nad WinRT běží v sandboxu, tj. nemůžou se navzájem ovlivňovat a například si mazat data (výjimkou jsou jen přesně definované

kontrakty) [9]. Veškeré operace, které by mohly trvat déle než 50ms, se dají z kódu volat pouze asynchronně. Pro stahování dat, komunikaci se sítí, ukládání souborů, nebo jen pro obyčejné zobrazení dialogového okna (obdoba *MessageBox*), jsou zde dostupné nové příkazy (v tomto případě *MessageDialog*). Pro jednodušší volání těchto asynchronních metod se používají nová klíčová slova *async* a *await*. Není ale k dispozici plný .NET Framework, jen určitá jeho podmnožina. Občas se mohou vyskytnou nesrovnalosti, že některé jmenné prostory se jmenují jinak, není k dispozici objekt typu *Thread*, místo *IsolatedStorage* je zde trochu jiné *Windows.Storage*, nebo že jazykový zdrojový soubor má odlišnou koncovku, než míval na Windows Phone 7.

Vzhled Metro, které WinRT implementuje, původně začalo ve Windows Phone 7. Nyní je již i ve Windows 8 a přináší zcela jiný vzhled ve srovnání s předchozími verzemi operačního systému Windows. XAML (Extensible Application Markup Language) je jazyk postavený na XML pro vývoj GUI a poskytuje jednotnou implementaci pro C++, C♯, VB a .NET framework [7]. Díky jazyka XAML je velice snadné vytvořit uživatelské rozhraní ve specifickém stylu Windows 8.

4 MonoGame

Informace obsažené v této kapitole byly čerpány především ze zdrojů [8] a [9], proto nebudou v této sekci již nadále uvedeny. MonoGame je open source framework pro tvorbu her a graficky náročných aplikací. Navazuje na XNA Framework od Microsoftu verze 4.X. Většina kódu je totožná s kódem z XNA. V jazyce C# lze pomocí MonoGame frameworku jednoduše vytvářet hry i aplikace nejen pro Windows 8. MonoGame je velice aktivní projekt a jak již bylo zmíněno výše, jedná se o open source implementaci. To s sebou nese jak výhody, tak i nevýhody. Mezi nevýhody se může zařadit fakt, že ne vždy je vše funkční, jak má být(občas se například může vyskytnout vyjímka *NotImplementedException*). Naopak výhodou je, že uživatel může přistoupit ke kódu a podle libosti si ho poupravit. Navíc je volně dostupné široká škála informací, protože vesměs MonoGame je totožné s XNA. Instalaci MonoGame lze provést dvěma způsoby. Může se použít instalátor, který nainstaluje šablony do Visual studia a další základní věci, avšak většinou neobsahuje nejaktuálnější verzi. Druhou možností je instalace přes zdrojové kódy, která není tak snadná jako při použití instalátoru, za to obsahuje vždy aktuální verzi. Jak nainstalovat MonoGame ze zdrojových kódů lze najít např. na oficiálním webu MonoGame frameworku.

4.1 Verze MonoGame

Jak již bylo zmíněno výše, použitím frameworku MonoGame se vývojář vystavuje nebezpečí, že může při vývoji narazit například na výjimku, že metoda není implementována, nebo mu nebude za určitých okolností třeba fungovat zvuk v jeho aplikaci apod. Vyvíjená aplikace byla implementována pomocí MonoGame frameworku verze 3.0.1 vydané 3. března 2013. Možná největším problémem této verze byl fakt, že nemá vlastní funkční content pipeline (viz sekce 4.4) a pro načítání jakéhokoliv obsahu bylo nutné jej nejdříve zkompilovat v XNA projektu a až poté převést výsledný .xnb soubor do aplikace v MonoGame. Pro tuto kompilaci tedy musel vývojář mít v počítači nainstalované XNA Game Studio a zároveň i Visual Studio 2010. Nová verze MonoGame 3.2 (oficiálně vydaná 7. dubna 2014) sice ještě vlastní content pipeline nemá, za to už ale existují dva způsoby jak se vyhnout podmínce mít nainstalované Visual Studio 2010 a přetahovat poté zkompilované soubory do požadovaného adresáře. Prvním je využití rozšíření, které umožňuje používat XNA i ve vyšších verzích Visual Studia než pouze 2010. Druhou možností přineslo samotné MonoGame a jedná se o nový projekt, nazývaný MonoGame Content Project, který lze vložit do společné solution a pomocí něho nahrávat a dále

používat veškerý podporovaný obsah (v následující tabulce jsou uvedeny přípony, jež by MonoGame mělo podporovat).

Typ obsahu	Typ souboru
3D modely	.x, .fbx
textury, obrázky	.bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm, .tga
audio	.xap, .wma, .mp3, .wav
fonty	.spritefont
efekty	.fx

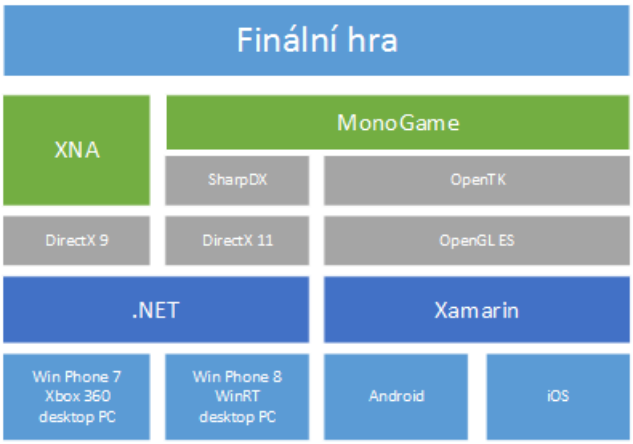
Tabulka 1: Podporované typy souborů

Nové změny v content pipeline avšak nejsou jediné. Nová verze 3.2 přišla s celkově 45 změnami. Nyní jsou už například dostupné šablony do Visual Studia 2013, takže se vývojář již nemusí omezovat pouze na nižší verze tohoto vývojového prostředí. Byla také přidána podpora dotyků pro Windows desktop aplikace a mnoho dalších nových změn. Jejich kompletní výčet lze najít opět na oficiální webu frameworku.

4.2 Multiplatformní MonoGame

Asi hlavní výhodou MonoGame frameworku je jeho multiplatformost. Momentálně podporuje platformy iOS, Android, Windows (OpenGL i DirectX), Mac OS X, Linux, Windows 8 Store, Windows Phone 8 a PlayStation Mobile. Stačí jednou napsat aplikaci a tu pak je možné spustit na jakékoli platformě. Avšak ne pro všechny platformy je tento Framework zdarma. Pokud je aplikace vyvíjena na Android, iOS, nebo OS X, je potřeba zakoupit licenci Xamarin (nazýváno také Mono touch nebo Mono droid). Xamarin vlastní i Xamarin studio, což je vývojové prostředí, ve kterém lze použít C# pro vývoj na Android a iOS. V tomto studiu lze pomocí designera vytvořit vlastní jednoduché uživatelské rozhraní s využitím jazyka XAML, nebo jej vytvořit pomocí samotného MonoGame. Existuje i verze zdarma, ale ta je omezena na pouhých 30Kb, což nepokryje ani samotné MonoGame. Zakoupená licence je doživotní, avšak pouze po dobu jednoho roku lze instalovat aktualizace. Pokud by po jednom roce bylo zapotřebí Xamarin aktualizovat, je nutné opět zakoupit novou licenci. Studentům a univerzitám je umožněno zakoupit licenci ve slevě.

Na operačním systému Android Xamarin zcela obchází Dalvika a Javu. Zpracovávání instrukcí se provádí zvlášť pomocí virtuálního stroje Mono CLR. Ve výsledku běží dva virtuální stroje zároveň. Dalvik je jedním a Mono druhým. Díky tomu může být vykonávání v Mono dokonce rychlejší než v samotné Jave. Aplikace jsou kompilovány just-in-time (za běhu aplikace) stejně jako s využitím Dalvika. Při použití Mono na Androidu není



Obrázek 6: Schéma MonoGame

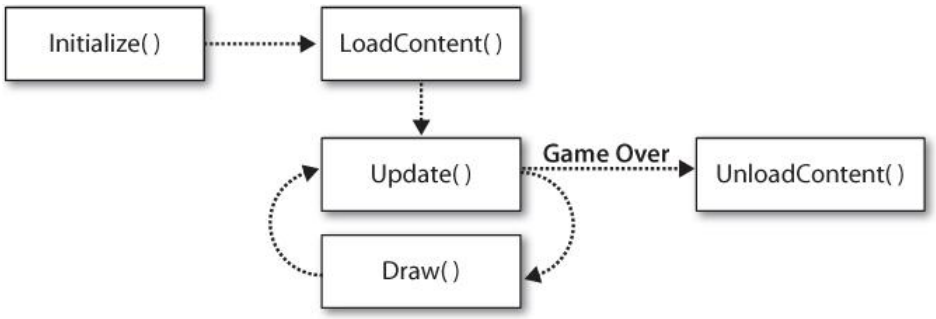
vývojář omezen funkčností. Například lze se připojit na knihovny od Javy, volat Javovské objekty apod.

U operačního systému iOS to funguje jinak než na androidu. Apple neumožnuje běh virtuálních strojů, takže se kód musí kompilovat už na začátku do strojového kódu. Po převedení se vytvoří nativní aplikace, která poté může být nahrána do zařízení. Opět je možné napojení na externí knihovny. V Business edici lze použít Xamarin pro Visual Studio a vyvíjet z něj. Xamarin poskytuje nové doplňky pro Visual Studio, které umožňují vývoj pro iOS (např. iOS panel nástrojů). V jiné edici než Business je uživatel nucen vlastnit zařízení Apple a vyvíjet na něm pomocí Xamarin studia.

4.3 Herní smyčka

Na obrázku č. 7 lze vidět herní smyčku, neboli game loop MonoGame frameworku. Stejně jako mnoho dalších věcí i herní smyčka je stejná jako v XNA.

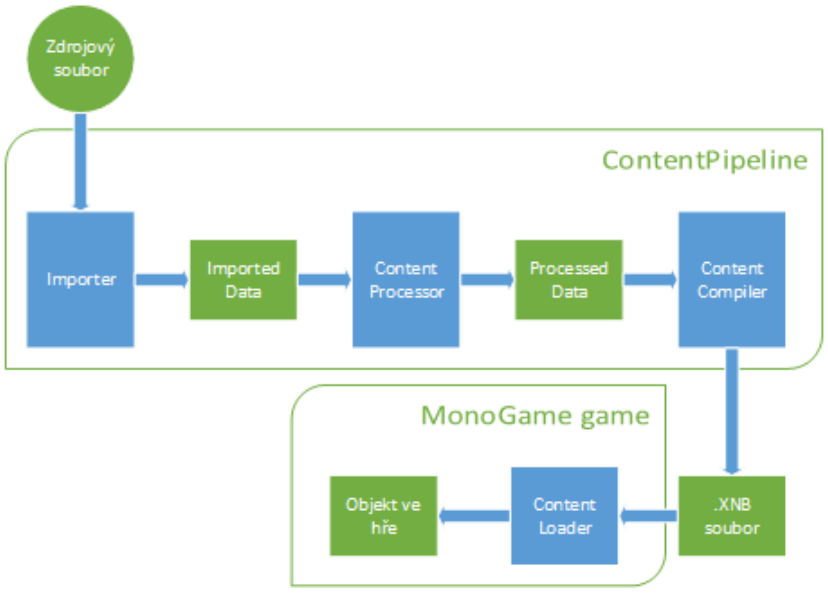
Jako první po zavolání konstruktoru hlavní třídy se provede metoda *Initialize()*. V ní si lze nastavit například pozice objektů, počet hráčů apod. Tato metoda slouží k inicializaci proměnných a provádí se pouze jednou na začátku hry stejně jako další metoda *LoadContent()*. Ta je určena k načtení obsahu hry, jako jsou například textury, modely ap. Kliknutí myší, táhnutí prstem a další vstupy do hry se řeší v metodě *Update()* společně s logikou hry a v metodě *Draw()* se vše vykreslí na obrazovku. Metody *Update()* a *Draw()* se provádí neustále dokola, pokud nedojde k ukončení hry. V takovém případě se nakonec zavolá metoda *UnloadContent()*, jež slouží k uvolnění obsahu, který byl načten jinak než pomocí třídy *ContentManager*.



Obrázek 7: Herní smyčka MonoGame frameworku [10]

4.4 Content Pipeline

Content Pipeline je velice užitečný doplněk MonoGame, který usnadňuje a zrychluje práci s načítáním zdrojových souborů. Poskytuje jednotnou metodu pro přístup k obsahu ve hře a také umožňuje distribuovat hry, bez nutnosti distribuce veškerého obsahu. Díky Content Pipeline je možné zabránit uživatelům přímému přístupu k modelům, texturám a dalšímu obsahu hry. Další výhodou může být fakt, že jsou odděleny povinnosti programátora od povinností autora obsahu. Když dojde ke kompilaci programu, všechen obsah jako textury, modely, fonty apod. projde Content Pipeline a uloží se do dočasného souboru.



Obrázek 8: Průběh souboru skrz Content Pipeline

Pokud poté načítáme konkrétní soubor v kódu, je nám vrácen z Content Pipeline jako

konkrétní herní objekt, jak lze vidět výše na obrázku č.8, který ukazuje průběh souboru skrz Content Pipeline. Průběh souboru se může rozdělit do čtyř fází. První je jeho načtení Importerem, jehož cílem je načíst data z DCC (Digital-Content Creation) souboru a převést je na objekt typu DOM (Document Object Model). Tento objekt již může přijmout Content Processor. Ten přijaté data převede z "hrubé" formy do více použitelné, jež je obvykle nějaký typ datové struktury. Následně tyto data převeme Content Compiler který z nich vytvoří binární soubor převážně s příponou .xnb. Poslední fází provádí Content Loader, který načte zkompilovaný .xnb soubor a vrátí již hotový objekt dále použitelný ve hře. Více informací o načítání obsahu je uvedeno v 5. kapitole.

To vše přišlo teprve v nedávné době. Pokud je použit k instalaci MonoGame instalátor, tak Content Pipeline nebude plně funkční a bude nutné kompilaci do .xnb souboru provádět v XNA projektu a následně až hotový soubor přesunout do MonoGame projektu. V této fázi by ale pořád obsah nešlo načíst. Ke správné funkčnosti je potřeba ještě nastavit v projektu vlastnosti .xnb souboru "Build Action" na "Content" a "Copy to Output Directory" na "Copy if never".

Jak již bylo zmíněno výše, Content Pipeline vytváří soubory .xnb, které ale mohou být velké a to s sebou může nést hned několik komplikací. Těmi mohou být například rychlost načítání, problém s pamětí apod. Při načítání textur lze Content Pipeline obejít a načítat je přímo ze streamu, což je ve výsledku rychlejší a u rozsáhlejších her to lze již poznat. Content Pipeline textury uchovává v cache paměti, takže pokud se použije načítání ze streamu, mělo by se cachování naimplementovat také, aby nedocházelo k neustálému načítání stejné textury. Dále by se mělo v události *DeviceReset* nastavit všechny textury na *null* a poté zavolat znovu *LoadContent()* metodu, protože při opuštění aplikace a znovu najetí do ní, textury sice jsou načteny, ale nezobrazí se. Zůstane po nich pouze černý obdélník.

4.5 Platformě závislé věci

Při práci s MonoGame je nutné řešit zvlášť platformě závislé věci (např. message boxy, logování chyb, systémové dialogy, apod.). Důležitou věcí je také samotné ukládání dat. Xamarin má pomocné funkce (např. IsolatedStorageFile), které slučují ukládání dat pro iOS a Android. Pro jiné platformy se běžně využívá metoda IsolatedStorageSettings, která pod nějakým určitým klíčem uloží objekt do konzistentního úložiště a při spuštění lze jednoduše tento objekt načíst a pracovat s ním. Komplikace se také vyskytují u DPI. Narozdíl od OpenGL ES, které automaticky sahá pro obsah do složky na základě toho, jaké je aktuální DPI, u MonoGame je nutno DPI zjišťovat v kódu a poté na jeho základě načítat správný obsah (to platí dokonce i u fontů) a provádět popřípadě přepočty pozic

objektů ve hře apod. Další komplikací při použití MonoGame na Windows 8 RT je fakt, že se musí zvlášť řešit vstup myši a dotyku (v každém volání metody *Update* dochází ke zjištění doteků a pokud k žádným nedošlo, tak se zjišťuje klik myší). Dále u této platformy nelze zapnout vyhlazování hran (antialiasing), protože WinRT tuto možnost nepodporuje.

4.6 Nahrání aplikace do Windows Storu

Pokud je již aplikace hotová a připravená k nahrání do Windows Storu (obchodu s aplikacemi pro Windows 8 RT), je zapotřebí ji ještě před samotným nahráním otestovat pomocí doplňku od Microsoftu zvaného Windows App Certification Kit (dále jen WACK). To z důvodu, že Windows Store aplikace procházejí přísnou certifikací. Také proto jsou tyto aplikace pro uživatele bezpečné a navíc se instalují do uživatelova profilu, takže nezanášejí systém jako celek. Nástroj WACK lze jednoduše stáhnout ze stránek Microsoftu určených pro vývojáře[2] a poté nainstalovat. Doplněk zjišťuje, zda aplikace splňuje všechny podmínky pro nahrání do Windows Storu pomocí mnoha testů. Pokud alespoň jedním z těchto testů neprojde, tak ani nahrání do Windows Storu se s největší pravděpodobností nepodaří. Nová WACK verze 3.3 (vydána 21. února 2014) byla aktualizována pro dopřednou certifikaci Windows Phone 8.1 aplikací a nyní podporuje následující typy aplikací:

- Desktopové aplikace pro Windows 8.1, Windows 8 a Windows 7
- Windows Store aplikace pro Windows 8.1 a Windows 8
- Windows Phone 8.1 aplikace

Další nutným kritériem pro vydání aplikace je registrace. Vývojář se musí zaregistrovat jako distributor aplikací Windows Store na webu[3]. Tato registrace je platná po dobu jednoho roku a je zpoplatněna částkou 49 USD pro fyzickou osobu a 99 USD pro právnickou osobu. Předplatitelé MSDN nebo studenti zaregistrovaní do projektu Dreamspark mají tuto registraci zdarma.

[2] adresa webu pro vývojáře: http://dev.windows.com/

[3] adresa pro registraci vývojáře: https://appdev.microsoft.com/StorePortals

5 Implementace aplikace

Implementovaná aplikace se skládá ze čtyř různých her pro trénování vybraných kognitivních funkcí. Lze ji ovládat jak myší a klávesnicí, tak i pomocí dotyků. Díky tomu je vhodná i pro trénování jemné motoriky. Aplikace byla vyvíjena pomocí frameworku MonoGame a je určená především pro tablety s operačním systémem Windows 8, avšak díky multiplatformnímu frameworku MonoGame lze po malých úpravách aplikaci použít i na jiných platformách.

Cube Game je první ze tří implementovaných her (obrázek č. 26). Tato hra využívá vizuální Forward Digit - Span test pro trénování a testování krátkodobé paměti. Jedná se o test, kdy uživateli je zobrazováno obecně postupně za sebou několik číslic a úkolem uživatele je tuto sekvenci zopakovat. Pro správnou funkčnost není nutné, aby byly zobrazovány pouze číslice [11]. Tento test je ve hře ztvárněn velkou krychlí uprostřed hrací plochy (obrazovky), která se otáčí o 90 stupňů. Úkolem uživatele je toto otáčení zopakovat ve správném směru a pořadí.

Další implementovanou hrou je *Model Game* (obrázek č. 27), jejíž cílem je rozvíjet prostorové vidění jedince. Úloha je postavená na testu mentální rotace. Uživateli jsou zobrazeny dva 3D obrazce a jeho úkolem je určit, zda jsou stejné. Pokud by tato úloha byla pro uživatele složitá, lze si v nastavení nastavit libovolný úhel a o ten poté hra umožňuje jedním z obrazců otáčet do všech směrů.

Třetí hrou je *N-back Game* (obrázek č. 28). Jedná se o implementaci stejnojmenného testu N-back, který je vhodný převážně pro pokročilé uživatele. Ve hře se rychle střídají obrázky a uživatel má za úkol na něj kliknout, objevil-li se právě před N kroky zpět. Tímto způsobem dochází k tréninku pracovní paměti a pozornosti.

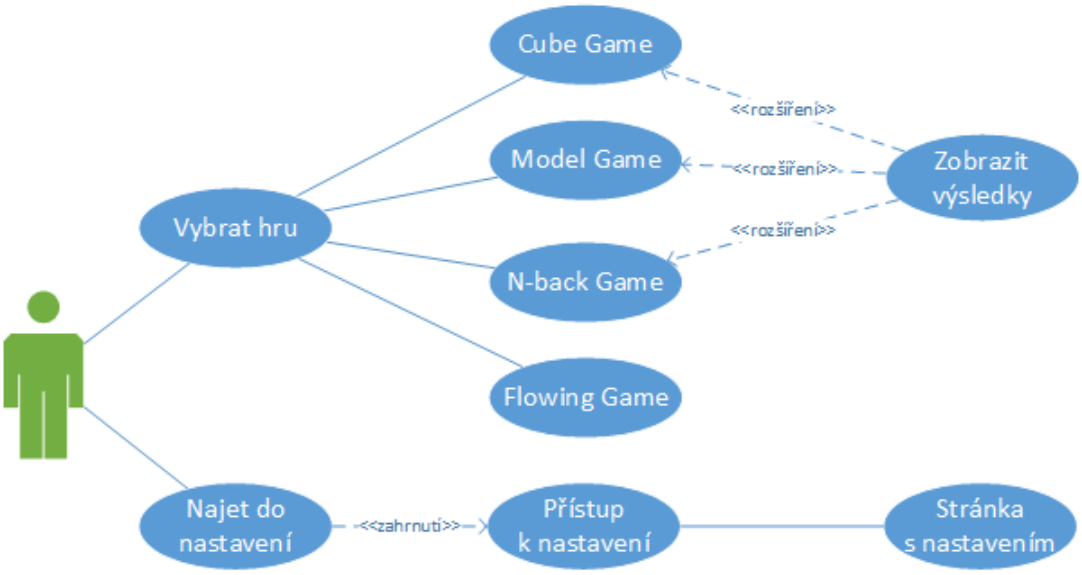
Poslední, čtvrtou úlohou v aplikaci je hra nazvaná *Flowing Cubes* (obrázek č. 29). Jedná se o 3D aplikaci, kde v prostoru poletují krychle a různě se odráží od stěn obrazovky (hranic hrací plochy). Uživatel má v této úloze za úkol zničit všechny poletující krychle. Ke zničení krychle dojde kliknutím, nebo dotekem na ní. U této úlohy se měří kolik krychlí dokáže uživatel zničit v co nejkratší době a s jakou přesností klikal. Tato přesnost se uvádí v procentech a je vypočtena podílem správných kliknutí k celkovému počtu kliknutí. Cílem této úlohy je rozvíjet u uživatelů především jemnou motoriku.

Po spuštění aplikace si může uživatel zvolit, zda chce přejít k výběru hry, nebo do nastavení jednotlivých her. V případě, že zvolí nastavení, zobrazí se uživateli požadovaná stránka uvedená na obrázku č. 30. Na této stránce s nastavením může uživatel nastavovat různé parametry jednotlivých úloh a tím určovat jejich složitost nebo vzhled. Pro první hru Cube Game lze nastavit barvu krychle (jestli jedna barva na všech stranách, nebo různé barvy), počet otáčení a jestli má být zapnuto osvětlení kostky. Pro druhou hru

Model Game lze nastavit barvy modelů, úhel povolené rotace, nebo osvětlení modelů. Pro třetí hru N-back je možno nastavit dobu zobrazení obrázků a samozřejmě hodnotu N, neboli počet kroků zpět, kdy byl daný obrázek zobrazen. Pro poslední čtvrtou hru Flowing Cubes lze nastavit počet létajících krychlí a rychlost, jakou se pohybují. Navíc lze ještě zapnout, popřípadě vypnout doprovodnou melodii, která jinak hraje v pozadí pro celou aplikaci.

Pokud uživatel zvolí možnost Hry, zobrazí se mu stránka se seznamem her a potřebných informací k nim (viz obrázek 25). Uživatel si zde může vybrat, jakou hru chce a přečíst si základní informace k ní (pravidla hry, co je cílem uživatele, k jakému účelu daná hra slouží, popřípadě jakou metodu využívá). Na této stránce si také danou hru může již pustit kliknutím na tlačítko "Hrát". Uživatel má možnost kdykoli v průběhu hry (kromě úlohy Flowing Cubes) si nechat zobrazit výsledky. Ty v aplikaci zůstávají uloženy až do doby, než dojde k jejímu ukončení. Tyto výsledky jsou v aplikaci uvedeny jak pomocí grafů, tak i textovým výčtem jednotlivých činností uživatele. Při hraní jakékoli hry má uživatel možnost se kdykoli vrátit zpět na seznam her a vybrat si hru jinou.

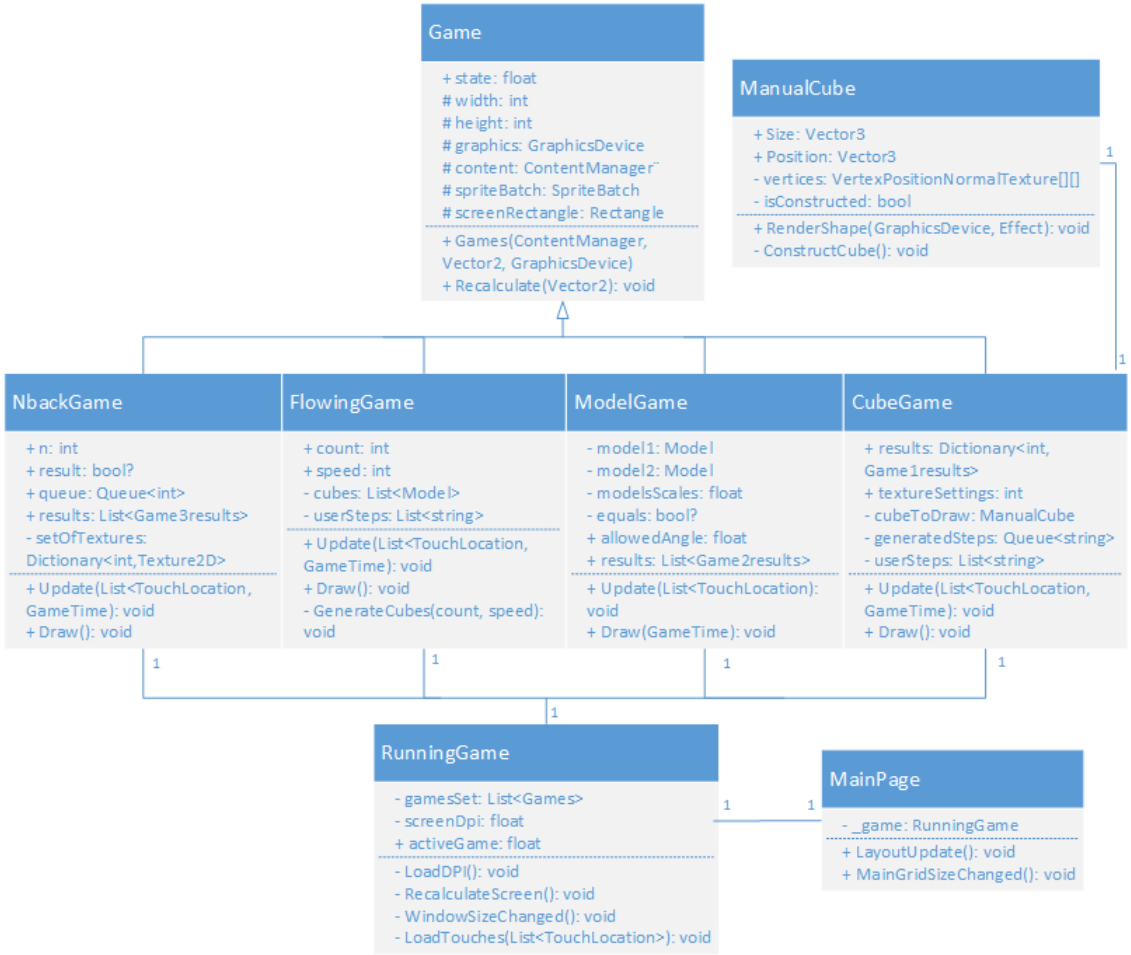
Diagram případů užití na obrázku č. 9 znázorňuje možné činnosti uživatele v aplikaci.



Obrázek 9: Diagram případu užití

5.1 Architektura aplikace

Při tvorbě uživatelského rozhraní aplikace bylo využito jazyka XAML a samotného MonoGame. Pomocí MonoGame se v aplikaci vykreslují jednotlivé hry a XAML je využíván pro veškeré ostatní uživatelské rozhraní. Místo funkce *main*, ve které by se běžně vytvářel *Game* objekt, se při použití MonoGame a XAMLu pro Window 8 využívá dvou souborů App.xaml a App.xaml.cs. Visual Studio automaticky vygeneruje třídu *App*, která dědí ze třídy *Application* a poskytuje první vstupní místo, kde lze již přidat inicializační kód. Dále lze asociovat handlery k událostem třídy *Application*, nebo přepsat některé z inicializačních metod jako například *OnLaunched()* apod. Na obrázku č.10 je uveden základní třídní diagram aplikace.



Obrázek 10: Třídní diagram pro vykreslování her

V aplikaci se ve třídě *App* vytváří instance třídy *MainPage*. Jedná se o code behind pro

úvodní XAML stránku *MainPage.xaml*. Dochází v ní ke spojení ovládacích prvků (tlačítka atp.) a vykreslování her pomocí MonoGame frameworku. Obsahuje převážně metody reagující na vybrané události daných ovládacích prvků. Metoda *GamePage_LayoutUpdated* je připojena k události *LayoutUpdated* a dochází v ní k nastavení, jaké ovládací prvky se zrovna mají zobrazit uživateli. To se provádí na základě atributu *activeGame* třídy *RunningGame*. Pokud se změní velikost obrazovky, dojde k vyvolání události *SizeChanged*, ke které je připojena metoda *MainGridSizeChanged*. Ta v závislosti na dané aktuální šířce a výšce přepočítá pozice všech ovládacích prvů.

Třída *RunningGame*, jejíž instance je vytvořena v konstruktoru třídy *MainPage*, obsahuje již předdefinované metody *Initialize*, *LoadContent*, *Update*, *Draw* a *UnloadContent* vytvářející herní smyčku. Tento základ se ve Visual Studiu vygeneruje automaticky po založení nového projektu. Tato třída již slouží k samotnému vykreslování her. Kromě vygenerovaných metod třída obsahuje i další doplňkové přidané metody jako jsou *LoadDpi* (stará se o načtení správného DPI), *RecalculateScreen* (po změně velikosti obrazovky přepočítává herní objekty), *windowSizeChanged* (tato metoda je připojena k události *Window.ClientSizeChanged* a reaguje na změnu velikosti obrazovky) a metoda *LoadTouches* (která zpracovává činnosti myší). V metodě *Inicialize* se vytváří instance samotných her a následně v metodě *Update* dochází k volání metod *Update* jednotlivých instancí. Taktéž se děje v metodě *Draw*, kde se volají metody *Draw* jednotlivých instancí. Každá herní třída dědí z nadřazené abstraktní třídy *Games* virtuální metodu *Recalculate*, jejíž parametrem je instance třídy *Vector2* představující aktuální šířku a výšku obrazovky. Na základě tohoto parametru se v metodě přepíše šířka a výška pro každou hru. Zde byl použit návrhový vzor Pozorovatel, díky kterému zareagují na změnu velikosti obrazovky všechny instance dědící ze třídy *Games*. Třída *RunningGame* obsahuje list všech instancí her a při vyvolání události *Window.ClientSizeChanged* se pro všechny instance zavolá právě metoda *Recalculate*. Dále třída *Games* obsahuje několik dalších parametrů, jež mají jednotlivé herní třídy společné (viz třídní diagram na obrázku číslo 10). Jak již bylo zmíněno výše, jednotlivé herní třídy obsahují metody *Update* a *Draw*. Ty slouží k aktualizaci dané hry a jejímu následnému vykreslení. Pro vykreslení krychle ve hře Cube Game využívá stejnojmenná třída třídu *ManualCube*. Ta obsahuje dvě důležité metody. První je *RenderCube*, po jejím zavolání se ověří, zda byla krychle již vytvořená a pokud ne, zavolá se metoda *ConstructCube*, která se postará o její vytvoření. Hry Model Game a Flowing Cubes využívají pro vykreslení 3D objektů jíž předem vytvořené modely a hra N-back Game je 2D, tedy na vykreslování se používají pouze bitmapy (o nich více v sekci 5.2)

5.2 Vykreslování 2D grafiky

Pro práci s 2D grafikou (bitmapy a fonty) má již framework MonoGame připravených mnoho tříd a jejich metod, díky kterým je následná samotná práce velice jednoduchá. V metodě *LoadContent* se požadovaný obsah načte, v metodě *Update* se poté provede veškerá logika (například posun, rotace...) a nakonec v metodě *Draw* se vše vykreslí.

5.2.1 Načítání fontů a textur

Načítání fontů a textur probíhá pomocí instance content třídy ContentManager. Ten umožňuje načítat jednoduše obsah z Content Pipeline pomocí jednotné metody *content.Load*, jak lze vidět v následující ukázce.

```
private Texture2D texture;
private SpriteFont spriteFont;

texture = content.Load<Texture2D>("Textures\\backgroundTexture");
spriteFont = content.Load<SpriteFont>("Fonts\\titleFont");
```

Výpis 1: Načtení fontů a textur

Při načítání fontů je nutné mít správně nastavené jeho vlastnosti (Build Action - Content a Copy to Output Directory - Copy if never). V jiném případě kompilátor zahlásí vyjímku "Could not load Fonts\SomeFont asset as a non - content file!". Jelikož metoda *content.Load* je generická, musí se při jejím volání specifikovat typ obsahu, který je načítán. Za názvy načítaných souborů se nepíší přípony, protože při kompilaci má všechen obsah příponu .xnb.

5.2.2 Sprity

Sprity jsou 2D bitmapy, které se vykreslují na obrazovku přímo, bez použití efektů, osvětlení apod. Běžně se používají například pro vykreslování textu a obrázků. Sprity se umisťují na obrazovku pomocí souřadnic, kde x-ová osa představuje šířku a y-ová osa výšku obrazovky. Souřadnice 0,0 je umístěna v levém horním rohu obrazovky. Pro vykreslení spritů se používá třída SpriteBatch, respektive její instance. Všechny prvky, které se mají zobrazit, se musí vykreslit mezi voláním metod *spriteBatch.Begin* a *spriteBatch.End*. Protože příprava grafického rozhraní je časově náročná, měly by se všechny sprity vykreslit najednou. Pro samotné vykreslení jednotlivých spritů se používá metoda *spriteBatch.Draw* viz výpis č.2

```
private Rectangle rectangle;
rectangle = new Rectangle(0, 0, width, height);

spriteBatch.Begin(SpriteBlendMode.AlphaBlend);
spriteBatch.Draw(texture, position, rectangle, color, rotation,
    origin, scale, effects, layerDepth);
spriteBatch.End();
```

Výpis 2: Vykreslení spritů

Uvedený parametr metody *spriteBatch.Begin* v ukázce povolí alpha blending (používané pro zobrazení alpha bitmapy, což je bitmapa, která má transparentní nebo semitransparentní pixely). Protože spriteBatch si může sám změnit nastavení vykreslování, lze u metody *spriteBatch.Begin* použít ještě parametr BlendState. Po jeho zadání (*SaveStateMode.SaveState*) se nastavení uloží. Následně dojde k vykreslení 2D grafiky a po zavolání *spriteBatch.End* se opět nastavení obnoví. Parametry metody *spriteBatch.Draw* jsou následující:

- **texture** – jedná se o texturu spritu (typ Texture2D)
- **position** – pozice spritu ve viewportu (typ Vector2)
- **rectangle** – obdélník (viewport), v rámci kterého se má sprite vykreslovat (typ Rectangle)
- **color** – barva osvětlení. Při volbě *Color.White*, bude sprite vykreslen beze změny. Po zvolení jiné barvy, se sprite vykreslí v jejím nádechu (typ Color)
- **rotation** – úhel rotace spritu v radiánech (typ float)
- **origin** – bod otáčení (typ Vector2)
- **scale** – násobek původní velikosti spritu zvlášť pro x-ovou a y-ovou osu (typ Vector2)
- **effects** – efekt použitý při vykreslování. Jedná se o horizontální či vertikální převrácení (typ SpriteEffect)
- **layerDepth** – vrstva, ve které je objekt umístěn od 0 – front do 1 – back (typ float)

V ukázce výpisu č.2 lze navíc vidět inicializaci objektu *rectangle*(neboli obdélníku), jenž se využívá pro vykreslení spritů. Jednotlivé sprity se vykreslují dovnitř obdélníku. Jeho první dva parametry udávají x-ovou a y-ovou souřadnici levého horního rohu a další dva jeho šířku a výšku.

5.2.3 Vektory

Jak si lze všimnou v ukázce výpisu č. 2, některé parametry metody *spriteBatch.Draw* jsou typu Vector2. Stejně jako XNA framework, tak i MonoGame umožňuje využívat struktury *Vector2*, *Vector3* a *Vector4*. Jelikož jsou vektory součástí téměř každé hry, tak velice usnadňují práci. Číslo za názvem *Vector* značí počet rozměrů. Jednotlivé vektory používají datový typ *float*. Dále obsahují mnoho užitečných metod a přetížené operátory, díky čemuž je velice snadné s vektory provádět mnoho matematických výpočtů (jako například sčítání, odčítání, násobení, dělní, měření vzdáleností dvou bodů apod.). Navíc tyto struktury obsahují pár již předdefinovaných vektorů (například *Vector2.Zero*).

5.2.4 Vykreslování fontů

Vykreslování fontů je velice podobné výše uvedenému vykreslování textur. Provádí se voláním metody *spriteBatch.DrawString*, jež taktéž musí být volána mezi metodami *spriteBatch.Begin* a *spriteBatch.End*.

```
spriteBatch.Begin();
spriteBatch.DrawString(font, text, position, color, rotation,
    origin, scale, effects, layerDepth);
spriteBatch.End();
```

Výpis 3: Vykreslení fontů

Tato metoda má většinu parametrů stejných jako metoda *spriteBatch.Draw* pro vykreslení textur, liší se pouze v prvních dvou. Prvním parametrem je font, kterým se má text vypsat (viz výpis č. 1) a druhým samotný text. Zbytek parametrů je již stejných.

5.3 Vykreslování 3D grafiky

Stejně jako pro 2D, má framework MonoGame připraveno mnoho tříd a metod i pro práci s 3D, díky kterým již není potřeba psát základní věci ručně.

5.3.1 Vytvoření 3D objektu

Základní věcí pro práci s 3D grafikou jsou takzvané vertexy. Vertex představuje bod v 3D prostoru. Jednotlivé body posléze lze spojovat do trojúhelníků, které dokáže grafická karta vykreslit na obrazovku. Jednotlivé body lze definovat pomocí několika struktur:

- **VertexPositionColor** – pozice vertexu a jeho barva

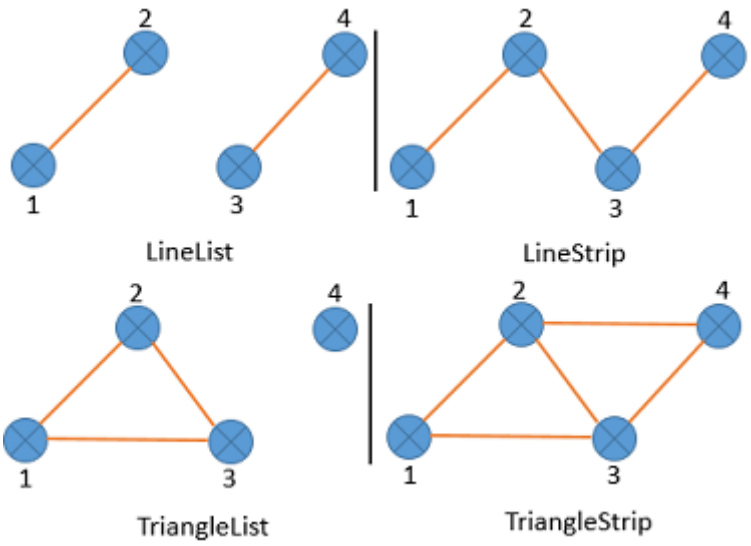
- **VertexPositionColorTexture** – pozice, barva a souřadnice pro texturu
- **VertexPositionTexture** – pozice a souřadnice pro texturu
- **VertexPositionNormalTexture** – pozice, normála a souřadnice pro texturu

V této práci byla pro potřeby vytvoření otexturované krychle použita poslední uvedená možnost *VertexPositionNormalTexture*. První parametr je typu Vector3 a udává pozici vertexu v 3D prostoru. Druhým je jeho normála typu Vector3. Ta je důležitá pro následné zobrazení osvětlení. Posledním třetím parametrem je proměnná *TextureCoordinate* typu *Vector2* udávající souřadnice textury. Tyto souřadnice pro textury se nazývají UV a nabývají hodnot od 0 do 1. Každý vertex tvořící trojúhelník tedy obsahuje svou UV souřadnici. Při vykreslování jednotlivých trojúhelníků se poté díky těmto souřadnicím sáhne na správné místo pro danou část textury. Spojením dvou trojúhelníků poté vznikne jedna správně otexturovaná strana krychle. Pro vytvoření celé krychle je třeba nadefinovat pozici, normálu dané stěny a UV souřadnice pro všechny vertexy. Jelikož jedna stěna se skládá ze dvou trojúhelníků, pro které je potřeba šesti bodů, je nutné tedy definovat celkem 36 vertexů.

5.3.2 Vykreslení 3D objektu

K vykreslování slouží výčtový typ *PrimitiveType*. Ten udává, jakým způsobem mají být jednotlivé body pospojovány. Spojovat je lze buď do trojúhelníku nebo čar. K dispozici jsou čtyři možnosti, kde dvě jsou určeny pro vykreslení čar a dvě pro trojúhelníky.

- **LineList** spojuje vertexy čárami a to vždy tak, že vezme jeden jako počáteční bod a druhý jako bod koncový.
- **LineStrip** taktéž spojuje vertexy čárami, avšak oproti LineListu každý bod spojí s jeho předchůdcem. Tím ve výsledku vznikne jedna klikatá čára .
- **TriangleList** vytváří trojúhelníky podobným způsobem jako LineList čáry. Bere body po třech a vytváří z nich samostatné trojúhelníky.
- **TriangleStrip** je obdobou LineStripu pro trojúhelníky. Po vytvoření prvního trojúhelníku se vezme další bod, který s posledníma dvěma vytvoří nový trojúhelník.



Obrázek 11: Ukázka jednotlivých primitivních typů

Pro případ, že by ani jeden z výše uvedených typů nevyhovoval, lze použít tzv. indexaci. Jedná se o pole typu *int*, ve kterém lze definovat indexy v poli vertexů které mají být spojeny.

```
GraphicsDevice.DrawUserPrimitives<VertexPositionNormalTexture>(primitiveType, vertexData,
    vertexOfffset, primitiveCount);
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionNormalTexture>(primitiveType,
    vertexData, vertexOfffset, numVertices, indexData, indexOfffset, primitiveCount);
```

Výpis 4: Vykreslování 3D objektu

Běžně se objekty vykreslují pomocí první z výše uvedených metod. Avšak pokud se použije indexace, musí se použít metoda druhá, kde její parametr *indexData* představuje právě ono pole typu *int* určené pro indexaci. Obě metody jsou generické, proto je potřeba při jejich volání specifikovat typ obsahu.

5.3.3 BasicEffect

BasicEffect je jedním ze základních efektů, který je součástí MonoGame. Dovede obsloužit základní vykreslení a pár snadných efektů, jako je například mlha. Každý efekt obsahuje 3 matice - World, View a Projection. Tyto matice určují, jak bude objekt vykreslen. Matice *World* umisťuje objekt do světa, posouvá s ním, rotuje a mění jeho měřítko. Matice *View* určuje odkud a kam se dívá kamera. Poslední matice *Projection* udává, jak má být vše vykresleno na obrazovku. Více o těchto třech maticích je uvedeno níže v této kapitole.

5.3.4 View space

View space je prostor kamery, jehož střed je v místě, kde se nachází kamera. Do tohoto prostoru se objekt dostane pomocí *View* matice, která slouží k definování pohledu odkud (první parametr) a kam (druhý parametr) se kamera dívá. V následujícím výpise č. 5 je ukázáno vytváření této matice pomocí třech parametrů. Prvním je pozice kamery, druhým místo, kam kamera směruje a poslední třetí parametr udává, jakým směrem má v jejím pohledu směr vzhůru (typicky se používá *Vector3.Up* - (0,1,0)).

```
BasicEffect effect;
effect.View = Matrix.CreateLookAt(cameraPosition, cameraTarget, cameraUpVector);
```

Výpis 5: Definování matice View

5.3.5 World space

Jedná se o prostor herního světa. Do tohoto prostoru se objekt transformuje pomocí matice *World*. Tato matice slouží k snadnému definování pozice objektu, jeho natočení a měřítku, v jakém se má vykreslit. Jelikož matice *World* může vzniknout násobením jednotlivých matic, které právě toto umožňují, je důležité tyto matice mezi sebou násobit ve správném pořadí (doporučené je uvedeno ve výpise č.6). To umožňuje objektem posunout, otočit a zvětšit, nebo zmenšit najednou. Pokud by například došlo nejdříve k posunu objektu a až posléze k rotaci, tak by se změnil střed rotace.

```
effect.World = Matrix.CreateScale(scale) *
               Matrix.CreateFromYawPitchRoll(yaw, pitch, roll)*
               Matrix.CreateTranslation(position);
```

Výpis 6: Definování matice World

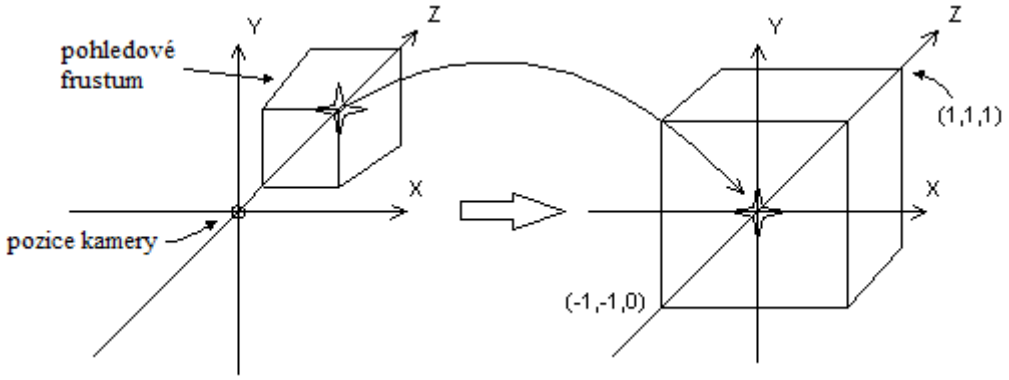
Posun objektu se provádí pomocí statické metody *Matrix.CreateTranslation*, jejíž parametrem je *Vector3*, který udává, kam se má daný objekt posunout. Pro rotaci objektu je možné použít následující metody:

- **CreateRotationX** – rotace kolem osy x o daný úhel
- **CreateRotationY** – rotace kolem osy y o daný úhel
- **CreateRotationZ** – rotace kolem osy z o daný úhel
- **CreateFromYawPitchRoll** – rotace kolem os y (Yaw), x (Pitch) a z (Roll) zároveň
- **CreateFromQuaternion** – vytvoří rotační matici ze struktury *Quaternion*

Metodou *Matrix.CreateScale* lze daný objekt zmenšit, nebo zvětšit (dojde k posunu bodů dále nebo blíže od středu prostoru).

5.3.6 Projection space

Tento prostor reprezentuje, co lze vidět na 2D monitoru. Jeho střed je uprostřed obrazovky. Nejedná se o rovinu, jak by se mohlo zdát, ale o kvádr, jež má rozměry 2x2x1. Do tohoto prostoru se objekt dostane pomocí projekční matice. Využívá se perspektivní projekce, která zobrazuje objekty tak, jak je lze vidět v realitě. Vzdálené jsou malé a ty, které jsou blízko se jeví velké.



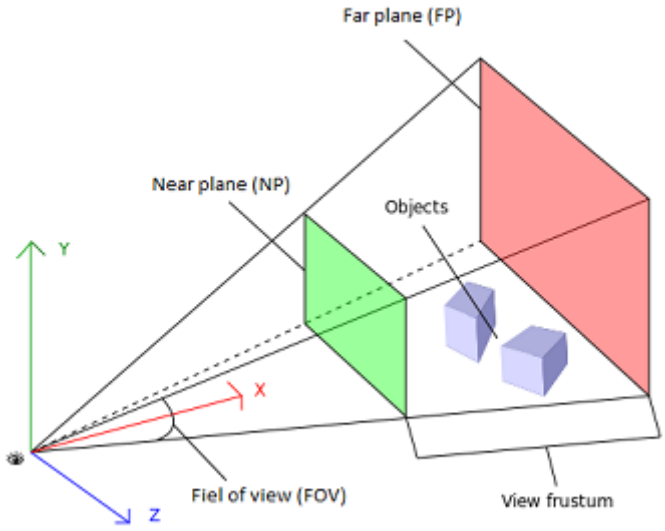
Obrázek 12: Převedení z pohledového prostoru do projekčního [13]

Na obrázku č.12 je vlevo uveden pohledový prostor (View space) a vpravo projekční (Projection space). To co je vidět v pohledovém prostoru, je obsažené v komolém jehlanu (pohledové frustum). Smyslem projekční matice je vzít bod, který je v prostoru tvořeném frustem a převést ho do prostoru tvořeném kvádrem. K vytvoření projekční matice slouží metoda uvedená ve výpise č. 7. Parametry této metody definují vzhled frusta.

```
effect.Projection = Matrix.CreatePerspectiveFieldOfView(fieldOfView, aspectRatio, near, far);
```

Výpis 7: Definování matice Projection

První uvedený parametr metody určuje, jak širokou výseč kamera sleduje (FOV na obrázku 13). Úhel se zadává v radiánech a běžně se uvádí 45 stupňů (pomocí statické metody *MathHelper.ToRadians* lze úhel zadávat i ve stupních). Druhým parametrem je poměr stran okna programu (neboli poměr stran plochy *Near plane*, nebo *Far plane* viz obr. 13). Následujícím parametrem je vzdálenost *Near plane* od kamery. Jedná se o pomyslnou plochu (začátek frusta), od které se začnou objekty vykreslovat. Vše co je před touto plochou není vidět. Tato vzdálenost se obvykle udává 1. Posledním parametrem je vzdálenost *Far plane* od kamery. Vše co je za touto pomyslnou plochou (koncem frusta) se už na obrazovku nevykreslí.



Obrázek 13: Pohledové frustum [14]

Pro vytvoření projekční matice lze využít i metodu *Matrix.CreateOrthographic*, jež provádí ortografickou projekci. Tato projekce zobrazí vzdálený objekt stejně velký, jako blízký objekt. Lze jí například použít u 2D vykreslování místo *spriteBatch*, který se o projekci stará sám.

5.3.7 Práce s modely

Model představuje v MonoGame libovolný trojrozměrný objekt. Je reprezentován třídou Model a jeho načítání do projektu probíhá stejně jako načítání jakéhokoliv jiného obsahu, tedy v metodě *LoadContent*.

```
Model model;
model = Content.Load<Model>("NazevModelu");
```

Výpis 8: Načítání modelů

V případě, že by v jednom souboru bylo více modelů obsahuje třída *Model* kolekci *Meshes* typu *ModelMesh*. Jednotlivý *ModelMesh* dále obsahuje kolekci *MeshPart* typu *ModelMeshPart*, což je třída obsahující potřebné informace k vykreslení části modelu (viz následující výpis, ve kterém jsou uvedeny nejdůležitější atributy).

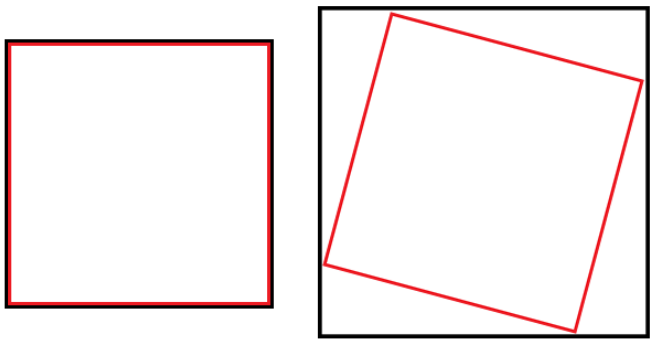
- **VertexBuffer** – Obsahuje všechny vertexy. Jeden vertex se může skládat z různých složek (minimálně z jeho pozice). Dále ale může obsahovat texturové koordináty, normálu...

- **IndexBuffer** – obsahuje indexy vertexů z vertex bufferu, tzn. pořadí v jakém se z nich skládají primitivy (většinou trojúhelníky)
- **StartIndex** – pozice v index bufferu, odkud se začnou číst vertexy
- **Effect** – načtením modelu přes content pipeline, se do této proměnné dostane instance třídy BasicEffect, se kterou lze dále pracovat

Při samotném vykreslování modelu se následně prochází jeho jednotlivé části (*Meshes*), kde se každé nastaví potřebné parametry a následně se po jedné vykreslují pomocí metody *Draw*.

5.4 Kolize 3D objektů

Základem kolizí jsou 3D objekty tzv. kolizní objekty, které obalují modely na obrazovce, samotné kolize zjednodušují a dělají je matematicky uskutečnitelnými v reálném čase. Modely v této aplikaci jsou pouze krychle, výpočetně a tvarově jednoduché objekty, ale pro další práci v aplikaci a možný výskyt složitějších modelů byly v práci vytvořeny metody pro obalování těchto modelů snazšími objekty. Obalením dochází ovšem ke ztrátě informace a v důsledku konečný výpočet není přesný, avšak výpočetní čas je mnohem kratší. Pro obalení se používají dva základní obrazce - koule (bounding sphere) a kvádr (bounding box). Pro výpočty je nejrychlejší výpočet kolize koule s koulí, kdy se spočítá pouze vzdálenost jejich středů. Kvádry již mají výpočet složitější, avšak existuje pro ně pár zjednodušujících pravidel. Tyto kvádry jsou osově orientované, jinak řečeno nejde s nimi rotovat a jejich hrany stále zůstávají rovnoběžné s osami herního světa, jak lze vidět na obrázku č. 14, kde napravo je ukázáno zvětšení bounding boxu (černý čtverec) při rotaci krychle (červeného čtverce).



Obrázek 14: Bounding box při rotaci

Jak lze vidět na obrázku, použití obalovacího kvádru pro rotující objekty není příliš vhodné. Pro tyto účely je lepší použít kouli, neboť ta při rotaci krychle zůstává pořád stejná.

5.4.1 Implementace kolizí

Pro řešení kolize je nejdříve nutné požadovaný model obalit jak je uvedeno výše. Obalení koulí ve frameworku MonoGame je velice snadné, protože ve třídě *ModelMesh* je již proměnná *BoundintSphere*, což je ve skutečnosti už samotná koule.

```
BoundingSphere sphere = new BoundingSphere(Vector3.Zero, 0);
foreach (ModelMesh mesh in model.Meshes)
{
    BoundingSphere transformed = mesh.BoundingSphere.Transform(transformace[mesh.
        ParentBone.Index]);
    sphere = BoundingSphere.CreateMerged(sphere, transformed);
}
```

Výpis 9: Vytvoření BoundingSphere

Princip získání koule je jednoduchý. Projdou se postupně všechny části modelu, kde každá tato část (*ModelMesh*) má již vytvořenou svou kouli. Tu je ovšem potřeba ještě transformovat. Toto se provádí uvedenou metodou ve výpise *Transform*, které se v parametru předává transformační matice pro danou část modelu. Další metodou *CreateMerged* se následně jednotlivé koule skládají do jedné, která je nakonec výslednou koulí obalující daný model. Pokud následně dojde ke změně měřítka, nebo posunu modelu, je potřeba pokaždé tuto kouli znovu transformovat.

```
Matrix transform = Matrix.CreateScale(Meritko) * Matrix.CreateTranslation(Pozice);
sphere = sphere.Transform(transform);
```

Výpis 10: Tansformace BoundingSphere

Stačí pouze vytvořit požadovanou transformační matici se správným měřítkem a pozicí a tu následně předat opět metodě *Transform*.

Při vytváření bounding boxu je postup poněkud složitější. Jelikož třída *ModelMesh* neobsahuje proměnnou *BoundingBox*, jak tomu bylo u koule. Třída *BoundingBox* sice obsahuje statickou metodu pro vytvoření obalového kvádru z obalové koule, avšak takto vytvořený výsledek není optimální. K použití obalového kvádru tedy bylo nutné si jej vytvořit manuálně pomocí bodů získaných z modelu. Princip vytvoření je opět velice snadný. Pomocí bodů, ze kterých se model skládá se najdou nejmenší souřadnice X, Y a Z. Následně se najdou i největší tyto souřadnice a vytvoří se z nich dva body. Z

těchto dvou bodů následně dojde k vytvoření požadovaného kvádru. Problémem může být fakt, že nalezení minima a maxima ze všech bodů může být trochu časově náročné. Je proto vhodné, toto použít pouze při prvním načtení modelu a následně jej pouze transformovat. Pro potřebu vytváření obalové krychle byla do práce implementována metoda *CalculateBoundingBox*, jejíž parametrem je model, který má být obalen. Metoda projde všechny části modelu, pro každou část nalezne její minima a maxima a porovná s dosavadními nalezenými minimy a maximy. Poté vybere z nich ty menší/větší hodnoty a tak to pokračuje dále pro všechny části modelu. Na závěr z nalezených dvou bodů vytvoří a vrátí vytvořenou instanci třídy *BoundingBox*, obdoba třídy *BoundingSphere* u koule (viz výše), která představuje právě požadovaný kvádr. Stejně jako u koule je potřeba při každé změně (měřítka, rotace nebo posunu) modelu, je potřeba obalový kvádr opět transformovat.

Po obalení modelu buďto kvádrem nebo koulí, je zjištění samotné kolize už triviální. MonoGame totiž pro obě třídy, jak *BoundingSphere*, tak i *BoundingBox* připravilo přetížené metody *Intersects*, sloužící právě pro zjištění zda je v kolizi z jiným kvádrem, či koulí (ukázka kódu ve výpise č.11).

```
if (sphere1.Intersects(sphere2))
   return true;
return false;
```

Výpis 11: Detekování kolize

V práci byly řešeny nejen vzájemné kolize modelů, ale i s hranicemi viditelné hrací plochy, aby nedošlo k posunu modelu mimo tyto hranice. Postup tohoto řešení je velice podobný výše uvedenému. Daný model se opět obalí buďto obalovým kvádrem, nebo koulí, aby bylo možné detekovat kolizi. To samé se provede i s pohledovým jehlanem, pro jehož obalení je v MonoGame vytvořena již třída zvaná *BoundingFrustum*. Vytvoření této instance a její následné použití v kódu pro detekci kolizí je ukázáno ve výpise níže (č. 12).

```
BoundingFrustum frustum = new BoundingFrustum(view * projection);
 if (frustum.Contains(bounding) == ContainmentType.Contains) { /*Do something*/ }
 else if (frustum.Contains(bounding) == ContainmentType.Disjoint) { /*Do something*/ }
 else if (frustum.Contains(bounding) == ContainmentType.Intersects) { /*Do something*/ }
```

Výpis 12: Tansformace BoundingSphere

Jak lze vidět ve výpise, pro vytvoření instance stačí pouze vynásobit pohledovou matici projekční a následně lze již použít její metodu *Contains*. Tato metoda je přetížená a lze jí předat instance *BoundingSphere*, *BoundingBox* a *Vector3*, neboli bod. Jejím výsledkem

nám je informace zda daná instance leží uvnitř pohledového jehlanu, mimo něj nebo zda je s ním v kolizi.

5.4.2 Vybírání

Pomocí tzv. vybírání (anglicky picking) lze získat informaci, nad kterým objektem (modelem) se zrovna nachází myš. Jinými slovy vybírání umožňuje vybrat nějaký 3D objekt pomocí myši, nebo doteku na obrazovku. Ve 2D aplikacích je vybírání objektů jednoduché, pouze se zkontroluje zda pozice myši je uvnitř například obdélníku, nebo kruhu. U 3D objektů je ale řešení vybírání trochu obtížnější. Slouží k tomu tzv. paprsek (anglicky ray), který je v MonoGame reprezentován stejnojmennou třídou *Ray*. Paprsek se definuje specifickým bodem, odkud vychází a směrem, kterým míří. Hlavní myšlenkou výběru je vytvořit paprsek vycházející z bližší roviny (na obrázku č. 13 zobrazena jako np - near plane) na vzdálenou rovinu pohledového jehlanu (na obrázku fp). Tyto dva body jsou definovány právě pozicí myši, nebo doteku a pomocí nich lze posléze vytvořit směr paprsku. Ve chvíli kdy dojde k jeho vytvoření, je již možné pomocí jeho metody *Intersects* detekovat kolize s instancemi tříd *BoundingBox*, *BoundingSphere* a *Plane*. V následujícím výpise je ukázáno, jak vytvořit takový paprsek na základě pozice myši a následně detekovat kolizi.

```
Vector3 nearPoint = new Vector3(mousePosition.x, mousePosition.y , 0);
Vector3 farPoint  = new Vector3(mousePosition.x, mousePosition.y , 1);

nearPoint = viewport.Unproject(nearPoint, camera.Projection, camera.View, Matrix.Identity);
farPoint  = viewport.Unproject(farPoint,  camera.Projection, camera.View, Matrix.Identity);

Vector3 direction  = farPoint  - nearPoint;
direction.Normalize();

Ray paprsek = new Ray(nearPoint, direction);
if (ray.Intersects(sphere) != null){ /* Is  in  collision */ }
```

Výpis 13: Tansformace BoundingSphere

Ve výpise si lze všimnout použití metody *Unproject*, která slouží k projekci bodu z 2D obrazovky na 3D pohledový jehlan. Po použití této metody se dané body stanou 3D a poté již lze vytvořit požadovaný směrový vektor, na jehož základě dojde k vytvoření samotného paprsku. Metoda *Intersects* vrací datový typ float, který je null, pokud v kolizi není. Pokud paprsek prochází daným modelem, vrací vzdálenost od paprsku k průniku.

5.5 Zvukové efekty v aplikaci

Zvukové efekty jsou nedílnou součástí naprosté většiny aplikací na nynějším trhu. Vzhledem k tomu, pro jakou cílovou skupinu je vyvíjená aplikace určena, bylo vhodné tyto efekty v aplikaci použít. Mohou vést například k lepšímu pochopení, rychlejšímu učení a zároveň se aplikace díky těmto efektům stává pro uživatele mnohem přívětivější.

Ve frameworku MonoGame jsou zvukové efekty představovány třídou *SoundEffect*. Aby bylo možné načíst konkrétní zvukový záznam do této třídy, musí se nejprve požadovaný zvuk nahrát do projektu. Toto se provádí podobně, jako například načítání fontů a jiného obsahu. Aby šel zvukový záznam v aplikaci přehrát, musí se do projektu vkládat v .xnb formátu. Toho lze dosáhnout kompilací daného zvuku v XNA Game Studiu, nebo využitím nového projektu MonoGame Content Project (viz kapitola 4.1), který dokáže zvukový záznam v běžných formátech převést do požadovaného .xnb souboru. Před samotnou kompilací se nesmí zapomenout na správné nastavení vlastností zvukového záznamu. Aby výsledný .xnb soubor bylo možné správně použít, musí být nastaveny vlastnosti Content Importer na MP3 Audio File, nebo WMA Audio File a vlastnost Content Processor musí být nastavena na Sound Effect. Poté lze soubor v pořádku zkompilovat. Při vkládání výsledného souboru do projektu se opět ještě musí nastavit vlastnosti, zda má být kopírován do výsledného adresáře a jaká akce se má provádět při kompilaci (jak již bylo zmíněno výše v kapitole 4.4).

Použití v aplikaci je pak velice podobné ostatnímu obsahu. Nejdříve se v metodě *LoadContent* daný zvukový záznam načte do proměnné a posléze s ním lze dále pracovat.

```
private SoundEffect effect;
effect = Content.Load<SoundEffect>("SoundDirectory/ConcreteSound");
effect.Play();
```

Výpis 14: Načítání zvukového záznamu

Jak lze vidět ve výpise č. 14, je nutné vytvořit instanci třídy *SoundEffect* zvlášť pro každý jednotlivý zvukový záznam, jež má být v aplikaci použit. Inicializace objektu se provádí stejně jako jiný obsah ve hře v metodě *LoadContent*. Takto vytvořený objekt umožňuje již přehrát daný záznam zavoláním metody *Play*, jak je uvedeno v ukázkovém výpise. Ovšem to je ale jediná možnost, co tato instance dovede. Tuto metodu je možné zavolat se třemi parametry uvedenými v následujícím výpise.

```
float volume = 1.0f;
float pitch = 0.0f;
float pan = 0.0f;
effect.Play(volume, pitch, pan);
```

Výpis 15: Parametrizované volání metody play()

Prvním parametrem je hlasitost. Ta nabývá hodnot od 0 do 1, kde 0 je nejnižší hlasitost (úplné ticho) a 1 je maximální hlasitost (hlasitost originálního zvukového záznamu). Dalším parametrem je výška. Jejím nastavením na -1 se sníží výšky o jednu oktávu, naopak +1 výšky o jednu oktávu zvýší. Pokud se nastaví výška na 0, znamená to, že bude stejná jako u originálního zvukového záznamu. Poslední třetí parametr se používá pro určení výstupu. Nabývá hodnot od -1 do 1, kde -1 znamená, že výstupem bude pouze levý reproduktor. Hodnota 1 naopak znamená že pouze pravý reproduktor. Pro použití sterea, neboli obou reproduktorů se nastavuje tato hodnota na 0.

Jak již ale bylo zmíněno výše, přehrání záznamu je jediné co instance třídy *SoundEffect* umí. Při běžném používání, je například zapotřebí záznam zastavit apod. K tomuto účelu je k dispozici ještě jedna třída. Ta se jmenuje *SoundEffectInstance* a vytváří se z výše uvedené instance třídy *SoundEffect*, jak lze vidět v následující ukázce č. 16.

```
SoundEffectInstance soundEffectInstance = effect.CreateInstance();
soundEffectInstance.Play();
soundEffectInstance.Pause();
soundEffectInstance.Stop();
```

Výpis 16: Vytvoření instance *SoundEffectInstance* z existujícího effektu

Tuto instanci lze využít stejně, jako tu minulou. Metoda *Play* se sice volá bez parametrů, ty ale lze nastavit přímo instanci. Navíc lze pomocí ní přehrávání zvukového efektu pozastavit, nebo úplně vypnout. Tato třída si poradí i s problémy jako jsou opakované přehrávání nebo zjišťování aktuálního stavu efektu (uvedeno ve výpise 17)). Pokud je v aplikaci například melodie, která hraje nepřetržitě, určitě je vhodné použít možnost nastavení smyčky přehrávání.

```
soundEffectInstance.IsLooped = true;
soundEffectInstance.Play();
if (soundEffectInstance.State == SoundState.Playing){
    soundEffectInstance.Stop();
}
```

Výpis 17: Ukázka možností třídy *SoundEffectInstance*

6 Použitelnost uživatelského rozhraní

Použitelnost uživatelského rozhraní je vlastnost aplikace, která představuje jednoduchost jejího užívání a snadnou naučitelnost zacházení s ní [15]. Vzhledem k cílové skupině uživatelů lze považovat dobrou použitelnost za prerekvizitu, kterou by vyvíjená aplikace měla splňovat, stejně jako jiné aplikace s podobným zaměřením. Vzhledem k tomu, že je určena pro osoby s určitým typem hendikepu, je vhodné využít tuto prerekvizitu, neboť může díky ní zabraňovat v nelibosti aplikace uživatelům. Špatné uživatelské rozhraní by mohlo vést například k odrazení od samotného používání aplikace a tím snížení jejího celkového účinku.

Mezioborová disciplína HCI (z anglického Human-computer interaction) se mimo jiné zabývá právě výše uvedenou použitelností [16]. Tato disciplína vznikla na přelomu 70. a 80. let 20. století za účelem zkoumání problematiky interakce a komunikace mezi počítačem a člověkem. Informační věda se v oblasti HCI zaměřuje především na tvorbu, design a testování použitelnosti aplikace tak, aby ve výsledku byla co nejjednodušší a nejintuitivnější pro cílovou skupinu uživatelů.

6.1 Studie použitelnosti

Pro tvorbu správného uživatelského rozhraní lze použít buď vlastního úsudku, nebo se řídit pokročilými metrikami použitelnosti právě výše uvedené disciplíny HCI. Použitím pokročilých metrik použitelnosti lze klasifikovat rozsáhlost problému, porovnávat mezi sebou dva produkty, zjistit skutečné využití aplikace apod. Tyto metriky s sebou přinášejí již konkrétní prokazatelná data, pomocí kterých lze aplikaci dále hodnotit. Lze je rozdělit do dvou základních skupin. První skupinou jsou pozorovatelné metriky (například zda daný uživatel splnil, či nesplnil určitou úlohu) a druhou skupinou jsou kvantifikovatelné (například že určité procento uživatelů splnilo, či nesplnilo dané zadání).

Pro studování použitelnosti existuje mnoho metod, přičemž jednou z nejzákladnějších a nejužitečnějších je tzv. uživatelské testování. Jak již název napovídá, jedná se o testování aplikace s konkrétní skupinou uživatelů, kteří přímo pracují s výslednou aplikací. Tato metoda se skládá ze třech následujících základních kroků:

- obstarání reprezentativní skupiny uživatelů, před kterou bude aplikace postavena,
- zadání reprezentativních úloh dané skupině,
- pozorování jejich řešení. Především klást důraz, kde uživatelé dělají chyby a mají s uživatelským rozhraním problémy, nebo naopak, kde uživatelé postupují správně.

.

Toto uživatelské testování bylo zrealizováno ve spolupráci se speciální školou v Poděbradech, kde reprezentativní skupina uživatelů plnila úlohy dle předem naplánovaných scénářů. Pomocí specializovaných pedagogů, kteří dohlíželi na toto testování, posléze došlo k vyhodnocení testované dílčí úlohy na základě pěti kvalitativních komponent. Dle Jakoba Nielsona, jednoho z hlavních představitelů oblasti HCI, je použitelnost uživatelského rozhraní definována právě těmito pěti komponentami [17].

- **Naučitelnost** – Jak snadné je pro uživatele se naučit základní úkoly při prvním setkání se s aplikací
- **Účinnost** – Jak rychle dokáží uživatelé plnit úkoly poté, co pochopili jejich zadání
- **Zapamatovatelnost** – Jak snadno mohou uživatelé znovuobnovit své znalosti, poté co nějakou dobu aplikaci nepoužívali
- **Chybovost** – Kolik chyb při používání aplikace uživatelé dělají, jak jsou tyto chyby závažné a jak snadno se z nich zotaví
- **Spokojenost** – Jak je příjemné používat uživatelské rozhraní aplikace

Specializovaní pedagogové ale nevyhodnocovali přímo samotnou použitelnost. Ve skutečnosti odpovídali na předem vytvořené otázky, které byly směrovány právě k výše uvedeným kvalitativním komponentám. Na základě odpovědí posléze došlo k vyhodnocení použitelnosti každé dílčí úlohy zvlášť a na závěr poté celé aplikace.

6.2 Testování použitelnosti

Jak již bylo zmíněno výše, testování probíhalo se speciální školou iSEN v Poděbradech, kde reprezentativní skupinou uživatelů byla pěti členná skupina dětí ve věku od 7 do 13 let, kteří trpěli převážně lehkým mentálním postižením. Testování probíhalo po dobu jednoho měsíce ve spolupráci se speciálními pedagogy, kteří na samotné testování dohlíželi a dle výsledků jednotlivých uživatelů odpovídali na předem připravené otázky:

- **Naučitelnost (0.15)** – Jak bylo pro dítě obtížné první použití aplikace?
- **Efektivita (0.3)** – Po seznámení s aplikací a po prvním použití, jak byste ohodnotili obtížnost dosažení cíle?
- **Zapamatovatelnost (0.15)** – Pokud se dítě vrací k aplikaci po delší přestávce (např. druhý den), uveďte s ohledem na délku přestávky schopnost opět aplikaci používat.

- **Chybovost (0.3)** – Jak často se dítě dopustí chyby při plnění úkolu aplikace?
- **Spokojenost (0.1)** – Jak byste ohodnotili spokojenost práce dítěte s aplikací?

Tyto otázky byly ve skutečnosti obrazem pěti kvalitativních komponent definovaných dle Jakoba Nielsona [17]. Na každou otázku mohli pedagogové odpovídat z celkem pěti předem vytvořených odpovědí, které byly reflektovány na stupnici od 0 od 100% s krokem po 25%. Tyto odpovědi a jejich reflektování lze vidět v tabulce č. 2.

Hodnota v %	Naučitelnost	Efektivita	Zapamatovatelnost	Spokojenost	Chybovost
100%	Snadné	Efektivní	Rychlé vybavení	Velmi Spokojené	Vůbec ne
75%	Snadné s obtížemi	Efektivní s obtížemi	Téměř rychlé vyb.	Spokojené	Občas
50%	Zvladatelné	Ef. s velkými obtížemi	Středně rychlé	Neutrální	Středně
25%	Obtížně zvladatelné	Málo efektivní	Pomalé	Spíše nespokojené	Často
0%	Velmi obtížné	Neefektivní	Celkově pomalé	Nespokojené	Velmi často

Tabulka 2: Reflektování odpovědí na stupnici 0-100%

Zároveň každé reflektované otázce modelu použitelnosti byla určena váha (číslo uvedené v závorce u každé z otázek). Toto číslo udává, jak je daná otázka objektivní. Například spokojenost může být ovlivněna i současnou náladou uživatele, proto má váhu pouze 0.1 (10%). Součet všech vah je roven 1. Výsledná použitelnost P'' dílčí úlohy aplikace pro jednoho testovaného respondenta se následně vypočte váženou sumou všech těchto vstupů (komponent dle modelu použitelnosti viz vzorec: 1).

$$P'' = \sum_{i=1}^{n} w_i x_i \tag{1}$$

Kde n je počet otázek, w_i je váha jednotlivé komponenty a x_i je odpověď spec. pedagoga pro danou komponentu reflektována na výše uvedenou stupnici. Jedná se o zjednodušený matematický popis modelu HCI použitelnosti dle Jakoba Nielsona. Každá komponenta bude mít tedy jen maximální jistý podíl na celkové použitelnosti, tedy např. chybovost bude zde max. 30%. Celková použitelnost P' jedné dílčí úlohy se následně spočte sumou použitelností každého respondenta (vzorec č. 2).

$$P' = \sum_{i=1}^{m} \frac{P''}{m} \tag{2}$$

Kde m představuje počet všech testovaných respondentů. Pro výpočet celkové použitelnosti aplikace P byl použit vzorec poměrové sumy č. 3.

$$P = \sum_{i=1}^{n} v_i \frac{P_i'}{n} \tag{3}$$

Kde P' jsou výše vypočtené použitelnosti jednotlivých dílčích úloh, n představuje počet úloh a v_i je váha každé této dílčí úlohy.

6.2.1 Výsledky testování

Na základě měsíčního testování aplikace došlo ke zpracování výsledků a následnému vyhodnocení. V následujících tabulkách jsou uvedeny výsledky, jak hodnotili specializovaní pedagogové práci respondenta s dílčími úlohami.

Respondent	Naučitelnost	Efektivita	Zapamatovatelnost	Spokojenost	Chybovost
1	Velmi obtížné	Neefektivní	Pomalé	Spíše nespokojené	Velmi často
2	Velmi obtížné	Málo efektivní	Pomalé	Spíše nespokojené	Často
3	Velmi obtížné	Neefektivní	Celkově pomalé	Nespokojené	Velmi často
4	Zvladatelné	Neefektivní	Celkově pomalé	Spíše nespokojené	Velmi často
5	Zvladatelné	Neefektivní	Celkově pomalé	Spíše nespokojené	Velmi často

Tabulka 3: Výsledky úlohy Cube Game

Odpovědi specializovaných pedagogů k první úloze Cube Game lze vidět v tabulce č. 3. Kromě samotných odpovědí na jednotlivé otázky pedagogové ještě zapisovali své poznatky k dané úloze. Dle těchto jejich poznatků, úloha pomáhá k posilování paměti, ale je příliš těžká pro jedince se střední mentální retardací a nejsou tedy schopni pochopit zadání úlohy i po několikanásobném předvedení. Bohužel v reakcích pedagogů nebyl žádný návrh na změnu v úloze a tím ji více přizpůsobit této cílové skupině uživatelů. Po reflektování odpovědí a použití výše uvedených vzorců vyšla výsledná použitelnost této úlohy pouze na 7,5%. Pouze pro respondenta č. 2 vyšla použitelnost této úlohy přes 20%. Ostatní respondenti dle spec. pedagogů nedosáhli ani hranice 10%.

Respondent	Naučitelnost	Efektivita	Zapamatovatelnost	Spokojenost	Chybovost
1	Zvladatelné	Efektivní s obtížemi	Středně rychlé	Neutrální	Středně
2	Snadné s obtížemi	Málo efektivní	Středně rychlé	Neutrální	Občas
3	Zvladatelné	Efektivní s obtížemi	Rychlé vybavení	Spokojené	Občas
4	Zvladatelné	Efektivní s obtížemi	Pomalé	Neutrální	Často
5	Zvladatelné	Efektivní s obtížemi	Pomalé	Neutrální	Často

Tabulka 4: Výsledky úlohy Model Game

V tabulce č. 4 jsou uvedeny odpovědi pedagogů ke druhé úloze Model Game. Již na první pohled si lze všimnou, že výsledky jsou mnohem lepší než u úlohy první. Dle

poznatků pedagogů z testování tato úloha přináší přínos převážně v oblastech zrakové pozornosti a rozvoje prostorové orientace. Jako problém u této úlohy viděli složitost zobrazovaných obrazců a grafické znázornění úspěchu či neúspěchu při plnění úkolu. Po aplikování výpočtu na tyto odpovědi je výsledná použitelnost této úlohy 55,75%. U respondenta č. 3 dosáhla dokonce 75% a nejnižší použitelnost byla u respondentů č. 4 a č. 5 a to 46,25%.

Respondent	Naučitelnost	Efektivita	Zapamatovatelnost	Spokojenost	Chybovost
1	Snadné s obtížemi	Efektivní s obtížemi	Téměř rychlé	Spokojené	Středně
2	Snadné	Efektivní s obtížemi	Rychlé vybavení	Spokojené	Občas
3	Snadné	Efektivní	Rychlé vybavení	Velmi spokojené	Občas
4	Zvladatelné	Efektivní s obtížemi	Téměř rychlé	Spokojené	Středně
5	Snadné	Efektivní	Středně rychlé	Neutrální	Občas

Tabulka 5: Výsledky úlohy N-back Game

Výsledky poslední třetí úlohy N-back Game byly zaneseny do tabulky č. 5. Tato úloha vyšla ze všech testovaných nejlépe a to s výslednou použitelností 77,75%. Nejvyšší použitelnosti úloha dosáhla u respondenta č. 3 a to 92,5%. Nejnižší byla naopak u respondenta č. 4 a to 66,25%. Dle poznatků pedagogů tato úloha přispívá k rozvoji zrakové paměti, zrakové pozornosti, soustředěnosti a pozornosti. Dle těchto specializovaných pedagogů je na nejlehčí obtížnost vhodná pro jedince se střední mentální retardací. Jako nedostatky této úlohy považovali absenci zvukového signálu podle úspěchu, či neúspěchu po kliknutí.

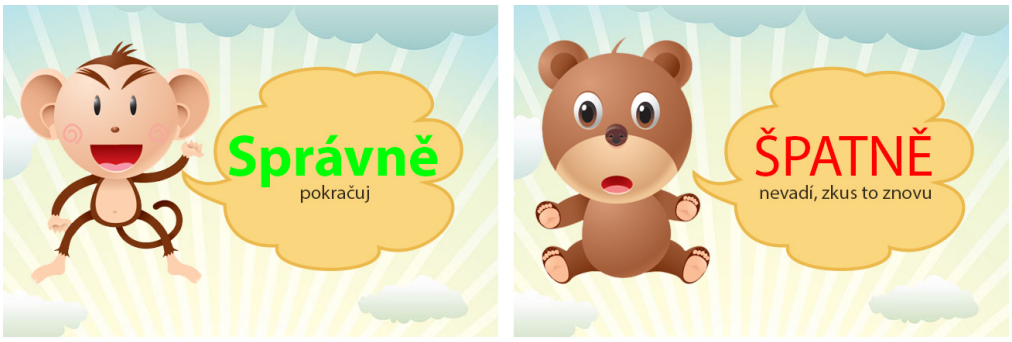
Po získání výsledků dílčích úloh došlo posléze k jejich dosazení do vzorce č. 3 a spočtení výsledné použitelnosti aplikace jako jednoho celku. Výsledná použitelnost dosáhla poté hodnoty 47%. Důvodem takto nízkého výsledku je především špatná cílová skupina, které byla aplikace předána k testování. Jak již v práci bylo zmíněno, aplikace byla testována dětmi s lehkou a střední mentální retardací, jejichž mentální věk nepřesahoval 5 let. To mělo za následek, že jedinci buď nedokázali pochopit zadání a nevěděli co se po nich chce (například u první úlohy *Cube Game*), nebo někteří jedinci nedokázali abstraktně přemýšlet a představit si celek jako ucelený tvar, jak tomu bylo u druhé úlohy *Model Game*. V důsledku nedostatku času ale již nebylo možné aplikaci předat k testování správné cílové skupině uživatelů.

6.3 Odraz testování použitelnosti na aplikaci

Vyjma první úlohy, měli specializovaní pedagogové, jež dohlíželi na testování, poznatky k použitelnosti. Tyto poznatky byly vzaty v potaz a došlo posléze ještě k několika úpravám v aplikaci tak, aby byly nedostatky odstraněny a mohla být aplikace více použitelná pro danou cílovou skupinu uživatelů.

Obrázek 15: Vyhodnoceni úkolu před testováním

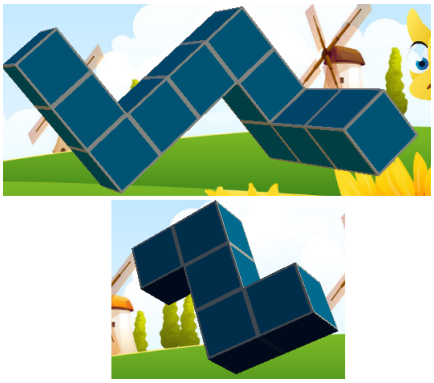
Obecně ve všech úlohách došlo ke změně zobrazení úspěchu, či neúspěchu při plnění úkolů. Stav ve kterém aplikace byla testována je zobrazen na obrázku č. 15. Hlavním problémem byl stejný obrázek pro úspěch i neúspěch při plnění úkolů. Respondenti tak nevěděli, zda úkol zdolali úspěšně nebo ne, a tak byl zvolen zvlášť obrázek pro úspěch i neúspěch. Zároveň navíc uvnitř bubliny byl zvýrazněn a zvětšen nápis pro lepší pochopení výsledku. Výslednou změnu lze vidět na obrázku č. 16.



Obrázek 16: Vyhodnoceni úkolu po testování

U úlohy Model Game měli spec. pedagogové poznatky ke složitosti zobrazovaných obrazců. Z tohoto důvodu vznikla nová sada obrazců, které by již neměli být pro tuto

cílovou skupinu tak složité. Na obrázku č. 17 lze vidět příklad rozdílu mezi původní verzí obrazce a upravenou, zjednodušenou verzí.



Obrázek 17: Ukázka zjednodušeného obrazce

V poslední třetí úloze N-back Game nedostatkem bylo zobrazení úspěchu, či neúspěchu při kliknutí. Pro vyřešení tohoto problému byly použity již upravené obrázky (viz obrázek č. 16). Navíc při plnění úkolu po kliknutí na obrázek se nyní hra pozastaví, aby měl uživatel více času pro zpozorování výsledku a také byly úloze přidány zvukové signály pro zdůraznění správného a špatného řešení úkolu. Dále nastavení této úlohy byla přidána volba pro možnost změnit dobu zobrazení jednotlivých obrázků a tím poskytnout uživateli více času pro zapamatování.

7 Výkonnostní testování aplikace

V rámci této práce došlo také k výkonnostnímu testování aplikace. Toto testování proběhlo v úloze Flowing Cubes. Z důvodu nedostupnosti zařízení pro účely testování proběhlo toto testování pouze na PC a tabletu. Parametry těchto dvou zařízení jsou uvedeny níže:

- Typ: notebook Acer Aspire 6930
- Operační systém: Windows 8.1 Pro
- Procesor: Intel Core 2 Duo P7450 2,13GHz
- Grafická karta: NVIDIA GeForce 9600M GT
- Typ: tablet Samsung ATIV Tab GT-P8510
- Operační systém: Windows RT 8.1
- Procesor: ARM - Qualcomm Snapdragon S4 Plus 1,5GHz
- Grafická karta: integrovaný grafický čip Adreno 225

Testování probíhalo způsobem zvyšování počtu vykreslujících se krychlí a při každém tomto zvýšení došlo k měření průměrného počtu snímků, jež se vykreslí na obrazovku za vteřinu (dále jen fps). Výsledky testování lze vidět v následující tabulce, kde v prvním řádku jsou uvedeny počty vykreslovaných krychlí a na dalších řádcích právě fps pro dané zařízení.

Typ zařízení	30	40	100	200	500	1000
Tablet	60	30	30	21	15	12
Notebook	60	60	60	60	60	55

Tabulka 6: Výsledky výkonnostního testování

Ve výsledcích si lze všimnout, že fps na tabletu se rapidně snížilo ze 60 na polovinu a na této hodnotě poté stagnovalo. K tomuto snížení dochází na mobilních zařízeních, pokud se metoda *Draw* provádí déle než 1/60s (16,67ms). Další vlastností frameworku MonoGame je nižší implicitní hodnota vykreslování snímků za sekundu u zařízení s operačním systémem iOS, Android a Windows Phone, která je nastavena rovnou na 30fps. Je tomu tak především z důvodu úspory energie a pro koncového uživatele je rozdíl mezi 30 a 60 snímky za sekundu téměř nerozeznatelný.

8 Závěr

Hlavním cílem této práce bylo především vyvinout aplikaci, která by byla použitelná pro rozvoj různých kognitivních dovedností a jemné motoriky pomocí jednoduchých cviků a úkolů. Vzhledem k nedostatku času, nebylo možné dlouhodobě aplikaci nasadit do reálného prostředí a pozorovat, zda u jednotlivých uživatelů dochází k nějakému rozvoji. V tomto důsledku nelze jednoznačně určit, zda vyvíjená aplikace svůj účel splnila, či nikoliv.

Vzhledem k cílové skupině uživatelů bylo nutné aplikaci správně navrhnout. Vývoj procházel různými fázemi, kdy v každé docházelo k výrazné změně uživatelského rozhraní (viz obrázky v příloze A.1). Důležitým faktorem při vývoji byla skutečnost, že většina jedinců z cílové skupiny uživatelů neumí číst. Z tohoto důvodu byla aplikace vyvíjena tak, aby byla pochopitelná právě bez nutnosti čtení. Například ke každému tlačítku jsou přiřazeny odpovídající obrázky. To ovšem neznamená, že aplikaci uživatel může obsluhovat sám. Aplikace se sice snaží být co nejjednodušší na pochopení, avšak i přes to je důležitá přítomnost asistenta uživatele (pro vysvětlení úloh a případné nastavení).

Pro další vývoj by bylo vhodné dodat aplikaci audio průvodce, jež by cílové skupině uživatelů velice usnadnil práci s aplikací. Dále by bylo vhodné vymyslet a doimplementovat systém odměn, který by motivoval uživatele a tím jej udržel v aplikaci po delší dobu. Přínosem pro aplikaci by také byly další dílčí úlohy zaměřené zase na jiné kognitivní funkce. Tuto diplomovou práci lze také využít pro další možný vývoj pomocí frameworku MonoGame, ať už jako výukový materiál, nebo základ pro herní engine.

Ondřej Gronych

9 Reference

[1] *Mentální retardace* [online]. Poslední revize 24. března 2015, [cit. 12.4.2015]
Dostupné z http://cs.wikipedia.org/wiki/Ment%C3%A1ln%C3%AD_retardace

[2] *Eda PLay* [online].c2013, [cit. 20.11.2014]
Dostupné z www.edaplay.cz

[3] KUŽNÍK, Jan. *Unikátní česká aplikace EDA PLAY pomáhá postiženým i zdravým dětem* [online]. c2013, [cit. 20.11.2014]
Dostupné z www.technet.idnes.cz/eda-play-rana-pece-0zq-/software.aspx?c=A130614_115715_software_kuz

[4] *Mind Games* [online]. c2013, [cit. 3.4.2015]
Dostupné z http://mindgames.mindware.mobi/

[5] *Cognitive Training* [online] [cit. 3.4.2015]
Dostupné z http://apps.microsoft.com/windows/en-nz/app/cognitive-training/0a6644d4-2093-41e9-884e-6f91199ef419

[6] *Competitive Brain Training* [online]. [cit. 3.4.2015]
Dostupné z https://play.google.com/store/apps/details?id=com.shimantech.funtastic&hl=en

[7] AHANI, Amir. *WinRT versus Win32 on Windows 8* [online].c2012 [cit. 8.4.2015]
Dostupné z blog.csharplearners.com/category/windows-8/

[8] *MonoGame* [online].c2015 [cit. 10.4.2015]
Dostupné z www.monogame.net

[9] SLAVÍČEK, Tomáš. *MonoGame: Vývoj her pro Windows 8, Android a iOS* [online].c2013 [cit. 10.4.2015]
Dostupné z http://smartmania.cz/clanky/monogame-vyvoj-her-pro-windows-8-android-a-ios-2-dil-4217

[10] STEWART, Kris. *CS 583 Lectures* [online].c2012 [cit. 10.4.2015]
Dostupné z http://www-rohan.sdsu.edu/ stewart/cs583/LearningXNA4_lects/

[11] *Memory span* [online]. Poslední revize 29.10.2014 [cit. 20.4.2015]
Dostupné z http://en.wikipedia.org/wiki/Memory_span

[12] IPPEL, Dennis. *UV coordinate basics* [online]. c2007 [cit. 17.4.2015]
Dostupné z www.rozengain.com/blog/2007/08/26/uv-coordinate-basics/

[13] *Tutoriály pro Xna* [online]. c2011 [cit. 10.4.2015]
Dostupné z xnaprototype.wordpress.com/

[14] SILVERMAN, Davis *Screen & viewport* [online]. c2013 [cit. 10.4.2015]
Dostupné z https://github.com/libgdx/libgdx/wiki/Screen-%26-viewport

[15] *Použitelnost* [online]. Poslední revize 27. února 2015, [cit. 3.3.2015]
Dostupné z http://cs.wikipedia.org/wiki/Pou%C5%BEitelnost

[16] *Human–computer interaction* [online]. Poslední revize 1. března 2015 [cit. 3.3.2015]
Dostupné z http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction

[17] NIELSEN, Jacob. *Introduction to Usability* [online]. c2012 [cit. 3.3.2015]
Dostupné z www.nngroup.com/articles/usability-101-introduction-to-usability/

A Ukázky aplikace

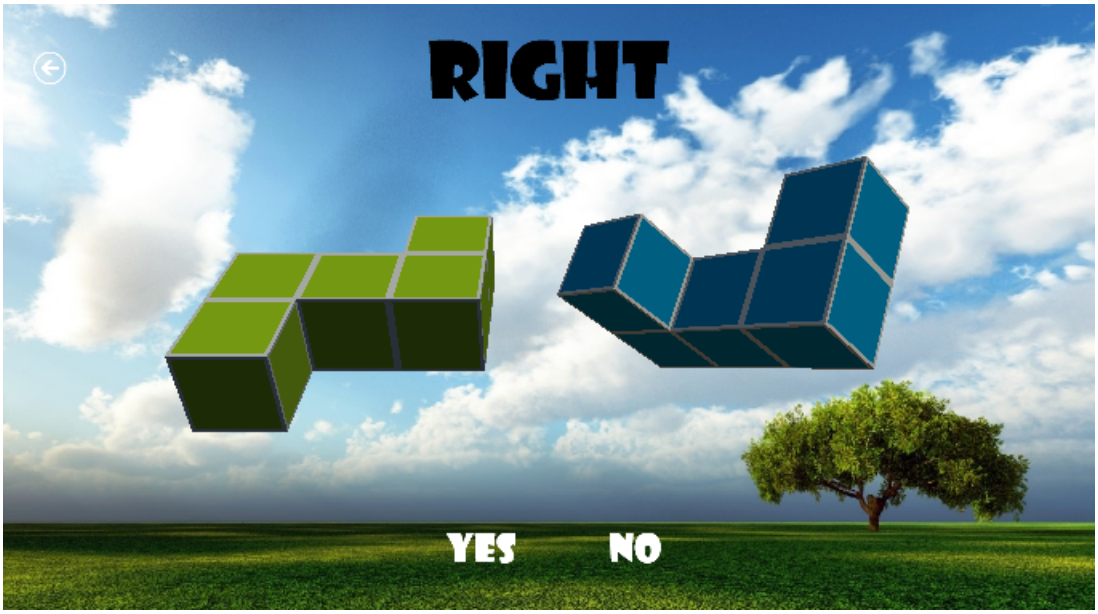
A.1 Postupný vývoj aplikace

Na následujících obrázcích je ukázáno, jak se postupně vyvíjelo uživatelské rozhraní úvodní obrazovky, seznamu her a jednotlivých úloh. Hra Flowing Cubes byla vyvíjena jako poslední a žádnou předchozí verzi neměla. Proto zde není uvedena. U stránky s nastavením a seznamem her se změnilo pouze pozadí a taktéž v této sekci nejsou zobrazeny.

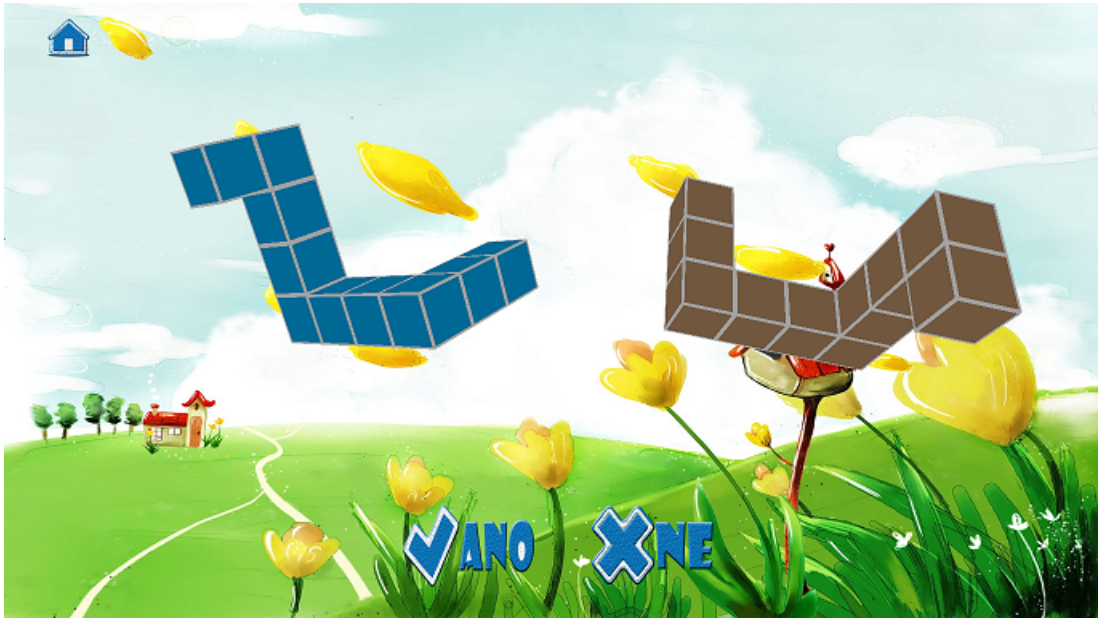
Obrázek 18: Úvodní stránka aplikace

Obrázek 19: Ukázka ze hry Cube Game - prvotní verze

Obrázek 20: Ukázka ze hry Cube Game - po prvních úpravách



Obrázek 21: Ukázka ze hry Model Game - prvotní verze



Obrázek 22: Ukázka ze hry Model Game - po prvních úpravách

Obrázek 23: Ukázka ze hry N-back

A.2 Ukázky finální verze aplikace

Na následujících obrázcích lze vidět finální stav aplikace. Konkrétně úvodní obrazovku, seznam možných úloh, jednotlivé úlohy a stránku s nastavením.

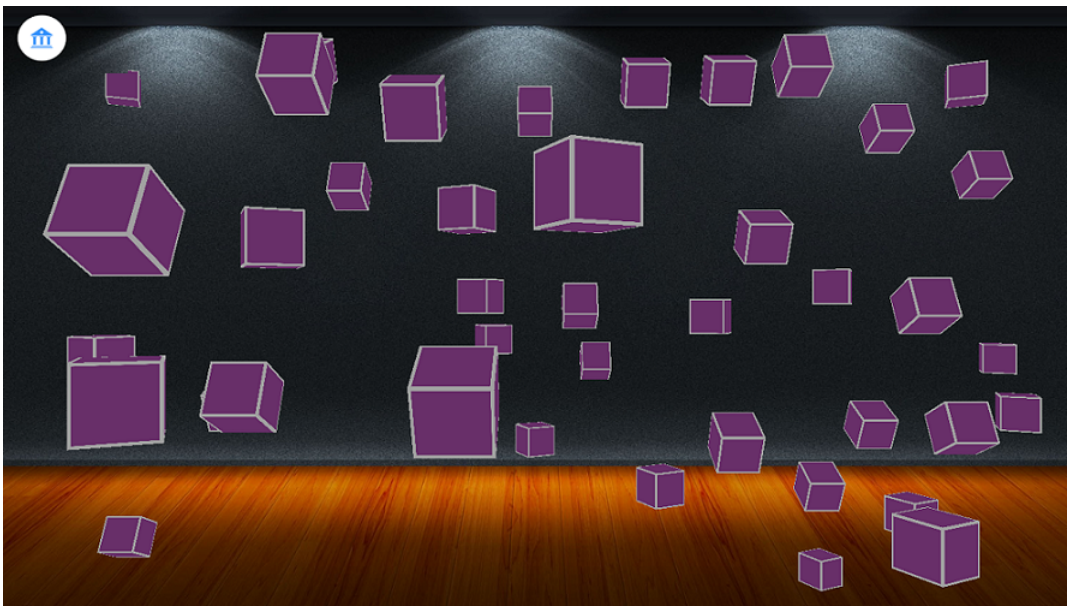
Obrázek 24: Úvodní stránka aplikace

Obrázek 25: Stránka se seznamem jednotlivých úloh

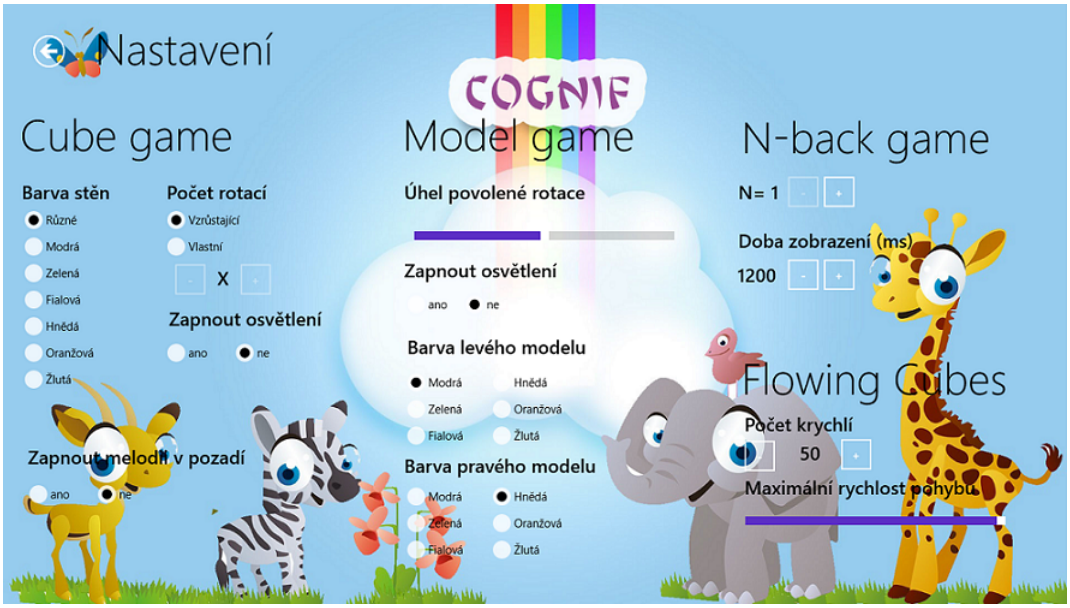
Obrázek 26: Ukázka ze hry Cube Game

Obrázek 27: Ukázka ze hry Model Game

Obrázek 28: Ukázka ze hry N-back Game



Obrázek 29: Ukázka ze hry Flowing Cubes



Obrázek 30: Stránka s nastavením

B Obsah přiloženého CD

Přiložené cd obsahuje dva soubory, které jsou uvedené v následující tabulce.

Soubor	Popis
DP_gro0015.pdf	Text diplomové práce
cognif.zip	Vyvíjená aplikace

Tabulka 7: Obsah CD